# Ecowatt

## FINAL PROJECT REPORT

Calvin College ENGR 340

**May 7, 2008**


**Team 10 Members**

**From Left:** Chris Michaels (EE), Jared Huffman (ME/EE), Brianna Bultema (ChE), and Achyut Shrestha (EE)

**© 2008, Calvin College, Ecowatt and Achyut Shrestha, Chris Michaels, Brianna Bultema, Jared Huffman**

# Table of Contents

1. Executive Summary ..... 8
2. Introduction ..... 9
3. Project Specification ..... 9
3.1. Problem Statement ..... 9
3.2. Project Objectives ..... 10
3.3. Project Requirement and Criteria ..... 11
3.4. Abbreviated Terms ..... 11
4. Research ..... 12
4.1. Microbial Fuel Cell ..... 12
4.2. MFC System Monitor ..... 12
4.3. Regulator Circuit Design ..... 13
4.4. Feeding Process ..... 13
5. Team Philosophy ..... 14
5.1. Christian Perspective on the Project ..... 14
5.2. Method of Approach ..... 14
5.2.1. Research ..... 14
5.2.2. Design ..... 14
5.2.3. Implementation ..... 15
5.3. Safety and Environmental Impact ..... 15
5.3.1. Electrical Impact ..... 16
5.3.2. Material Impact ..... 16
5.3.3. Chemical/Biological Impact ..... 16
6. Proposed Solution ..... 18
6.1. Project Management ..... 18
6.1.1. Team Organization ..... 18
6.1.2. Team Member Bios ..... 19
6.1.3. Team Member Roles Description ..... 20
6.1.4. Schedule ..... 21
6.1.5. Contingency Plan for Individual Failure/Absence ..... 27

6.2. Cost Estimates for Building Prototype ..... 28
6.2.1. Electrical ..... 28
6.2.2. Case ..... 28
6.2.3. Biology Materials ..... 29
6.3. Business Plan for Full Scale Production of Product ..... 29
6.3.1. Marketing ..... 29
6.3.2. Basic Production Process ..... 29
6.3.3. Budget ..... 30
6.4. Design ..... 30
6.4.1. Anode Cube ..... 30
6.4.2. Feeding ..... 32
6.4.3. Case Design ..... 33
6.4.4. Microbial Fuel Cells ..... 44
6.4.5. Fuel Cell Arrangement ..... 48
6.4.6. Power Management ..... 49
6.4.7. Feeding ..... 49
6.4.8. Regulation ..... 50
6.4.9. USB Power Switch ..... 56
6.4.10. Monitoring System ..... 56
6.4.11. Fuel Cell Arrangement Alternatives and Analysis ..... 72
6.5. Test Results ..... 73
6.5.1. Monitor ..... 73
6.5.2. Regulation ..... 74
7. Conclusion ..... 79
7.1. Lessons Learned ..... 79
7.2. Future Work ..... 79
8. Acknowledgements ..... 80
9. References ..... 81
10. Appendices ..... 83

10.1. Appendix I - Biobattery Assembly Instructions ..... 84
10.2. Appendix II- Bladder Filling Procedure ..... 87
10.3. Appendix III - MFC Testing Data ..... 88
10.4. Appendix IV - Product Information Sheet of ATCC® 53774 ..... 93
10.5. Appendix V - Membrane Testing Data ..... 94
10.6. Appendix VI - Nafion 117 Information sheet ..... 95
10.7. Appendix VII - Software Tools ..... 96
10.8. Appendix VIII – ATMega169 Specifications ..... 98
10.8.1. Features ..... 98
10.8.2. Schematic ..... 99
10.9. Appendix IX – Monitor Source Code (Written in C) ..... 100
10.10. Appendix X – Regulator Test Results ..... 124

## Table of Figures

Figure 1 - Method of Approach ..... 15
Figure 2 - Team Organization Chart ..... 18
Figure 3 - Gantt Chart of Fall Semester Plan ..... 23
Figure 4 - Gantt Chart of Fall Semester Actual ..... 24
Figure 5 - Gantt Chart of Spring Semester Plan ..... 25
Figure 6 - System Interaction ..... 30
Figure 7 - Anode Cube Model ..... 31
Figure 8 - Feeding System Process Flow Chart ..... 33
Figure 9 - Full Case - Closed ..... 35
Figure 10 - Case with Food Bladders ..... 36
Figure 11 - Case with Anodes Cubes ..... 37
Figure 12 - Case with Support Rails ..... 38

Figure 13 - Component: Waste Chamber (and Feed Bladder Chamber) ..... 39
Figure 14 - Component: Cathode Body ..... 40
Figure 15 - Component: Support Rail ..... 41
Figure 16 - Component: Anode Cube ..... 42
Figure 17 - Component: Feed Bladder ..... 43
Figure 18 - Schematic of Microbial Fuel Cell ..... 44
Figure 19 - Cellophane Membrane Testing ..... 47
Figure 20 - Nafion117 Membrane Testing ..... 48
Figure 21 - Power Management Block Diagram ..... 49
Figure 22 - Regulator Circuit Diagram ..... 55
Figure 23 - MAX1607 Circuit Layout ..... 56
Figure 24 - Monitor Block Diagram ..... 59
Figure 25 - AVR Butterfly Kit ..... 60
Figure 26 - Voltage Port Schematic ..... 61
Figure 27 - Temperature Sensor Schematic ..... 62
Figure 28 - USI Port Schematic ..... 63
Figure 29 - LCD Schematic ..... 64
Figure 30 - Port B Schematic ..... 65
Figure 31 - MFC Power Check Function ..... 66
Figure 32 - Temperature Check Function ..... 67
Figure 33 - Monitor Flow Chart ..... 69
Figure 34 - Main function of Monitor Firmware Part 1 of 2 ..... 70
Figure 35 - Main function of Monitor Firmware Part 2 of 2 ..... 71
Figure 36 - Input Voltage vs Output Voltage ..... 76

Figure 37 - Input Current vs Output Current ..... 76

Figure 38 - Input Power vs Output Power ..... 77

Figure 39 - Efficiency with Variable Input Voltage ..... 77

Figure 40 - Load Resistance vs Output Current ..... 78

Figure 41 - AVR Studio 4 IDE ..... 96

Figure 42 - WinAVR IDE ..... 97

Figure 43 - ATMega169 Schematic ..... 99

## Table of Tables

Table 1 - Abbreviated Terms ..... 11

Table 2 - Chemical Concentration in Feed ..... 17

Table 3 - Team Member Roles ..... 21

Table 4 - Fall 2007 Major Milestones and Dates Accomplished ..... 26

Table 5 - Spring 2008 Major Milestones and Dates Accomplished ..... 26

Table 6 - Case Design Bill of Materials ..... 35

Table 7 - Monitor Voltage Level Display ..... 66

Table 8 - Monitor Temperature Level Display ..... 67

# 1. Executive Summary

The Ecowatt Biobattery successfully addresses and overcomes many of the previous design problems inhibiting the proliferation of Microbial Fuel Cells (MFCs) as a viable energy alternative. The research showed that MFCs have been implemented in previous studies at the university level, but no viable products have come to fruition. Ecowatt testing early on proved the MFC concept – achieving an average closed circuit voltage output of 0.3V for over a week without feeding. An assortment of sources indicated that more power can be achieved with a more efficient proton-exchange membrane. Further testing proved the functionality of a new cost-effective membrane: cellophane. The electrodes are assembled adjacent to the membrane in a sandwich method for optimum proton exchange efficiency, and the sandwiches are arranged in a cube structure to allow for easier feeding of multiple cells and to maximize volumetric efficiency. Even with all of these improvements, the Biobattery still requires over 90 cubes to provide proper USB power; it fits roughly into a 2' X 2' X 3' cubic space and weighs about 500lbs when filled with water. Further areas to improve on in the future would be efficiency of the electrodes, implementation of Platonized electrodes to eliminate the need for a water cathode chamber (and about 55% of the weight), and research into more effective fabrication methods to prevent leaks and allow for better utilization of cube surface area.

Over the course of this project, the team has learned valuable lessons. First of all, it is important to maintain and adhere to a schedule. Ecowatt was successful in this respect for the first semester and then fell behind during the prototyping and testing stage. Moreover, prototyping and testing are much more time consuming than originally anticipated, and long lead times on part orders can drastically impact a team schedule. The team also learned the importance of enlisting the help of others in the industry and other universities. Assistance from researchers and those in the industry was paramount in the acquiring of parts far outside the meager Ecowatt budget. Finally, the team experienced the joy of seeing a project grow from brainstormed ideas to a completed design over the course of the year. In light of this, the project is clearly successful.

# 2. Introduction

Ecowatt is a four member senior design group of senior engineers at Calvin College who seek to have a positive electrical impact on the world. This team of four engineers includes sections of three of the engineering concentrations offered at Calvin. Two members of the team are electrical engineering concentration (Achyut Shrestha and Chris Michaels), one is a chemical concentration (Brianna Bultema) and one member is a hybrid concentration of mechanical and electrical (Jared Huffman). This interdisciplinary engineering team has been working on this design project through two semesters and two courses, ENGR 339 and ENGR 340. Ecowatt chose their senior design project due to their interests in environmental and power generation engineering and their desire to be a service to others and God.

# 3. Project Specification

## 3.1. Problem Statement

Conventional batteries are widely used and may be bought at a reasonably low price. These batteries can power electronics for a couple of weeks up to a year, and they are thrown away. Some batteries are even rechargeable and last longer but with decreased performance for every subsequent charge.

Unfortunately, conventional batteries are not as low cost as they seem. The materials used in conventional batteries are toxic, not easily recyclable and harmful to the environment. In an attempt to minimize their impact, battery processing centers have been implemented, but the process is expensive and time-consuming, and the vast majority of batteries are simply discarded into the “trash.”

Ecowatt is a team that is called to answer the problems caused by conventional batteries by developing Microbial Fuel Cells into a power source that is portable, consistent, and reliable. The power source must have a reduced environmental impact by not using toxic or harmful chemicals, be more easily disposed than conventional batteries, and have a longer lifespan. It must also be easy to operate and provide a reasonable amount of stable power in usable, nominal, USB Standard values, without requiring frequent or expensive maintenance.

## 3.2. Project Objectives

For this project, Ecowatt will design and create a prototype for a microbial fuel cell or "Biobattery." The term "Biobattery," for the purposes of this project, refers to a device which generates power by Geobacter, a type of microbial organism. The Biobattery designed by Ecowatt will consist of a Microbial Fuel Cell (MFC) system comprising a Feed and Water System, an Output Regulator, and a MFC Monitoring System.

All features of the case design must be reasonably robust and leak-resistant. The case must be easy to carry and intuitive to use. The Biobattery should be contained in an attractive package that appeals to the target market.

Each MFC contains two compartments separated by a proton exchange membrane that generate an electric charge as a result of the eating behavior of the microbes. MFC design will also include specifications for the appropriate electrodes and a proton exchange membrane between the two solutions.

The Feed and Water System will manage the flow of feed solution into the MFC as well as the flow of water solution out of the MFC. The Feed and Water System also includes design of the MFC chambers and a Tank Refilling Station.

The Output Regulator will ensure a stable power output at standard battery voltage and consistent current whenever the MFC is functioning.

A MFC Monitoring System will communicate to a user by relaying important information such as the voltage level of the MFC and feeding time for Geobacter.

## 3.3. Project Requirement and Criteria

There are minimum requirements that must be fulfilled in order to meet the objectives of the project. So the design must meet following criteria for this project to be successful.

- The microbial fuel cell array must provide at least 5 volts.
- The user must be able to feed the Geobacter without any unreasonable difficulty.
- There must be a feed indicator for letting the user know that it is time to feed Geobacter.
- The circuit implemented for monitoring and regulating the voltage must use minimum possible power from the MFCs.
- In case the circuit fails, the Biobattery should still provide a voltage output and also let user know about the failure.
- The Biobattery case must protect the Biobattery contents from the outside world and protect the outside world from the Biobattery contents.
- The terminals from the Biobattery must be easily accessible for a user to power a device.
- The circuit must protect itself and the connected device from any voltage spikes, so it must provide a stable voltage and not cause any damage to connected device when functioning properly. The circuit must also ensure that it does not inadvertently damage the MFC itself.
- Geobacter should not outgrow its living environment nor die off when operated under standard conditions.
- Finally, the system must be portable, appealing, and user-friendly.

## 3.4. Abbreviated Terms

**Table 1 - Abbreviated Terms**

| | |
|---|---|
| ADC | Analog to Digital Converter |
| IC | Integrated Circuit |
| IDE | Integrated Development Environment |
| LCD | Liquid Crystal Display |
| LED | Light Emitting Diode |
| MFC | Microbial Fuel Cell |
| MOSFET | Metal Oxide Semiconducting Field Effect Transistor |
| RoHS | Reduction of Hazardous Substances |
| USB | Universal Serial Bus |

# 4. Research

Engineering is not designing in a void of information. As such, prior work was analyzed and used as a stepping stool for improvement on the current application. Other approaches have been found in literature, but none have had the application that Ecowatt is designing.

## 4.1. Microbial Fuel Cell

Microbial fuel cells (MFC) have been in research papers since the early 1970s. Three main types have received the most attention in the last 15 years: heterotropic, photoheterotropic and sediment cells. Heterotropic and photoheterotropic cells use suspended bacterial cultures to produce current. Sediment cells are historically made with sediment found from the local body of water, using an electrode and cathode (electrically conductive metals) placed in the sediment and water above the sediment. Heterotropic cells have consistently provided higher power to surface area ratios which allow for smaller electrode areas.

## 4.2. MFC System Monitor

The focus of the research was to find what exactly needs monitoring and what solutions are available to monitor the system. The MFC is a dynamic system. The power generation from Geobacter depends upon several biological and chemical factors. Nevertheless, the Monitor design for this bio-battery system is in the interest of the functionality of the system and not the inner workings of Geobacter. Designing a monitoring system with focus on MFC environment will lead towards experimental science, which is not the objective of the Monitor.

The MFC produces voltage, which varies overtime. Monitoring the voltage produced by the MFC gives an idea about power generation. The voltage output of the MFC also depends upon the feedstock available to the Geobacter and the removal of the waste from the MFC. Therefore, it is necessary to check if the feed is low or the waste level is high. When connecting a load to MFC, the current drawn by that load influences the power generated. If a load draws too much current, the Geobacter cannot keep up and the efficiency of the system goes down dramatically. The research shows that it is necessary to monitor the current withdrawn from the system. Temperature also heavily influences how active the Geobacter is in the given environment. It cannot produce electricity when the surrounding temperature is out of range. The optimal temperature range is 20-35 Celsius.

After finding the variables that need monitoring, the research focused on how to monitor on those variables. One can monitor the variables with either analog devices or digital devices. Analog devices are easier to implement. A mechanical voltmeter can measure the voltage output from MFC, a thermometer can measure the temperature, and over-current protection can utilize a fuse. These analog devices are all individual components. Hence, they

will work regardless of other components function. Digital devices are more robust. Digital devices provide for a better monitoring system. Integrating all the different sensors together can provide meaningful information. However, they are somewhat complicated to implement. To implement the Monitor using digital devices will at least require a temperature sensor, an ADC, a micro-controller, and an LCD for display.

## 4.3. Regulator Circuit Design

Much of the preliminary research for the voltage regulator consisted of internet searches at sites such as Wikipedia, scientific databases, and scientific journals. Research was limited to topologies that were possible to use for regulating and conditioning the Biobattery's specific power output. When numbers for the approximate output of the Biobattery became available, much better design alternatives became available as well. The design alternatives reflect the design criteria in this case.

## 4.4. Feeding Process

Microbial fuel cells contain bacteria that must have a sufficient supply of "food" to survive. In fact, the power generation performance of a microbial fuel cell is dependent on food supply. Since the bacteria must live in an anaerobic environment, the method of adding the food to the cell solution requires finesse.

The food supply can be added to the cell solution periodically in controlled amounts through a process called batch feeding. At feeding time, the solution must be maintained while the food supply is added and the waste solution is simultaneously removed. Unfortunately, batch feeding must be done at relatively frequent intervals, and the power output can drop significantly starting about one week after last feeding.

To solve the problems associated with batch feeding, researchers have employed continuous feeding. Continuous feeding maintains a steady flow of food supply into the cell and waste solution out of the microbial fuel cell. Requiring little maintenance or upkeep. On the other hand, continuous feeding systems are complicated and expensive to implement. Pumps would be necessary to continually draw the food supply into the microbial fuel cell; the power for such pumps would use more energy than the Biobattery could possibly generate, rendering them unfeasible.

To optimize the benefits and overcome the drawbacks to both batch and continuous feeding processes, a combination of the two methods is implemented in the design. The batch-fed continuous feeding process will be discussed in detail in the Design section.

# 5. Team Philosophy

## 5.1. Christian Perspective on the Project

The ministry of Jesus Christ is critical to the Ecowatt team as an inspiration for change. Ecowatt is making its way into an industry that, for the most part, seems concerned only with the design factors of low cost and high performance. Seeking to make a change, the design team is dedicated to three values based primarily on the example of love shown in the life Jesus Christ.

Any project from Ecowatt must adhere to the Fundamental Ecowatt Values of Stewardship, Trust, and Justice. For viability's sake, each project must use global resources responsibly; the Biobattery project will provide an option for portable power that does not use corrosive chemicals. In addition, dependability and reliability are a must. The Biobattery project will provide a stable power supply in a convenient, easy-to-use package. Ecowatt will publicly disclose any issues of general public or environmental safety. Finally, the Biobattery project is an expression of Justice; the rights of non-users and users are a critical consideration in the design process. When considering materials, environmental and health impacts are separately weighted. Even in production, the effect on every person is the subject of careful deliberation.

## 5.2. Method of Approach

This project uses the basic approach of researching, designing and implementing for a successful completion of the project. Although there are three distinct phases in the project, they overlap with each other over the course of the project.

### 5.2.1. Research

The first phase of the approach is research. Research is of utmost importance to our project. Microbial fuel cell is a recent technology and there is a lot of research going on in this field. We have been able to obtain plenty of information about MFCs.

### 5.2.2. Design

The next phase of the approach is design. After researching necessary materials, the next step to do is develop a design of the product that we will build. The design is mostly done on paper and computer. The design itself is divided into other steps. The first design is just the block diagram of each component in the system so there is a basic understanding of how the system should function. Then next step is to take each component of the system and blow it into each individual sub-system. Each sub-system has a block diagram of its own to show how it functions. Then each components of the sub-system is designed according to the required specification.

### 5.2.3. Implementation

The last phase of the approach is implementation, or prototyping. Each component of the sub-system is implemented either with computer model/simulation or a prototype. Without implementation there is no way to know if the design actually works.

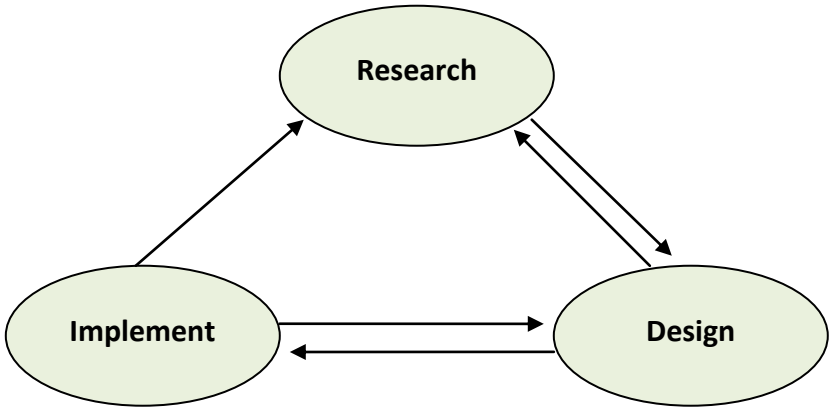



Figure 1 - Method of Approach

Although the approach seems linear, the phases overlap (Figure 1 - Method of Approach). The implementation will not always work as designed or planned, so we need to go back to our design and readjust the changes. Sometimes we might even have to go back to our research and find out what we overlooked. Many times design has to be revised based on new findings and researching, so we are constantly changing design for improvement, so the approach is actually cyclic.

## 5.3. Safety and Environmental Impact

There are consequences to our actions that impact environment, sometimes negative even unintentional. Many products in the market claim to be green but are produced and manufactured without the environment in consideration. So not only a product should be environmentally friendly but it is also preferable that the environment itself where the product is made be green. So the other aspect of our design is to ensure that the process used for making MFC itself does not impact the environment in negative ways. This means that the methodology used also must have minimal and positive effects on the environment.

Equally important is the safety of the device and anyone who is operating the device. One way to ensure safety is providing appropriate labels for caution and warning. However the more effective way is to educate a user about the correct usage of product. Safety must also be practiced while producing the product itself. Still the safety of the MFC must also be considered so that foreign objects do not intrude into the system causing inadvertent effects.

### 5.3.1. Electrical Impact

The safety and environment is of utmost concern for the electrical components used in the system. The components used must be RoHS compliant and lead free. The electrical device is one of the hardest materials to recycle. They also have a massive carbon footprint. Since the size of the components matter, using surface mount parts to reduce the carbon footprint as much as possible is important. Electrical devices are also power hungry. Whenever any component is not in use, it is better to turn it off or at least leave it in an idle state so the power consumption decreases dramatically. When the lifetime of an electrical device is over, some components still are reusable. Recycling printed circuit boards can also go to extract the metal present in them. However, recycling electrical parts are expensive. The safety of electrical devices needs great care, especially when these operate near a moist environment. Any liquid spill can cause the circuit to malfunction, which will lead to replacing the part. In order to avoid unintentional accidents, the electrical components need careful packaging so no foreign material can destroy the parts. Therefore, a robust design will take safety and environment concerns in mind and prevent any forcible accidents. The design must also include the safety of a user operating the device.

### 5.3.2. Material Impact

The materials involved in the design are carefully chosen to minimize the environmental impact while meeting the design goals. Modularity allows most of the parts to be reusable, and the Feed Bladders are to be accepted by the manufacturer for sterilization and refilling. For more information on material selection, see the Case Design Section on page 33.

### 5.3.3. Chemical/Biological Impact

The feed solution that *Geobacter* consumes contains chemicals including sodium acetate, sodium citrate, Potassium Chloride, and others. Sodium acetate is one of the products of the well-know fizzing reaction of baking soda and vinegar used in kid's volcano projects. The two other products are carbon dioxide (the fizzing gas) and water. Other uses for sodium acetate include food flavoring for salt and vinegar chips, as a food preservative and in hand warmers. Our sodium acetate is purchased from a chemical supplier and is 99.9% pure. The feed solution is a dilute mixture as seen in table 2.

**Table 2 - Chemical Concentration in Feed**

| Name | mol/L | Industrial Usage |
|---|---|---|
| Sodium Acetate | 0.0500 | Food Additive, Hand Warmers |
| Sodium Citrate | 0.0466 | Food Additive, Buffer, Medical |
| Ammonium Chloride | 0.0047 | Food Additive, Shampoo, Snow treatment, Soddering |
| Potassium Chloride | 0.0013 | Fertilizer, Food Processing, Substitution for Table Salt |
| Sodium bicarbonate | 0.0298 | Baking (baking soda), deodorizer, cosmetic use, cleaning |
| Monosodium phosphate | 0.0050 | Food Additive |

# 6. Proposed Solution

## 6.1. Project Management

### 6.1.1. Team Organization

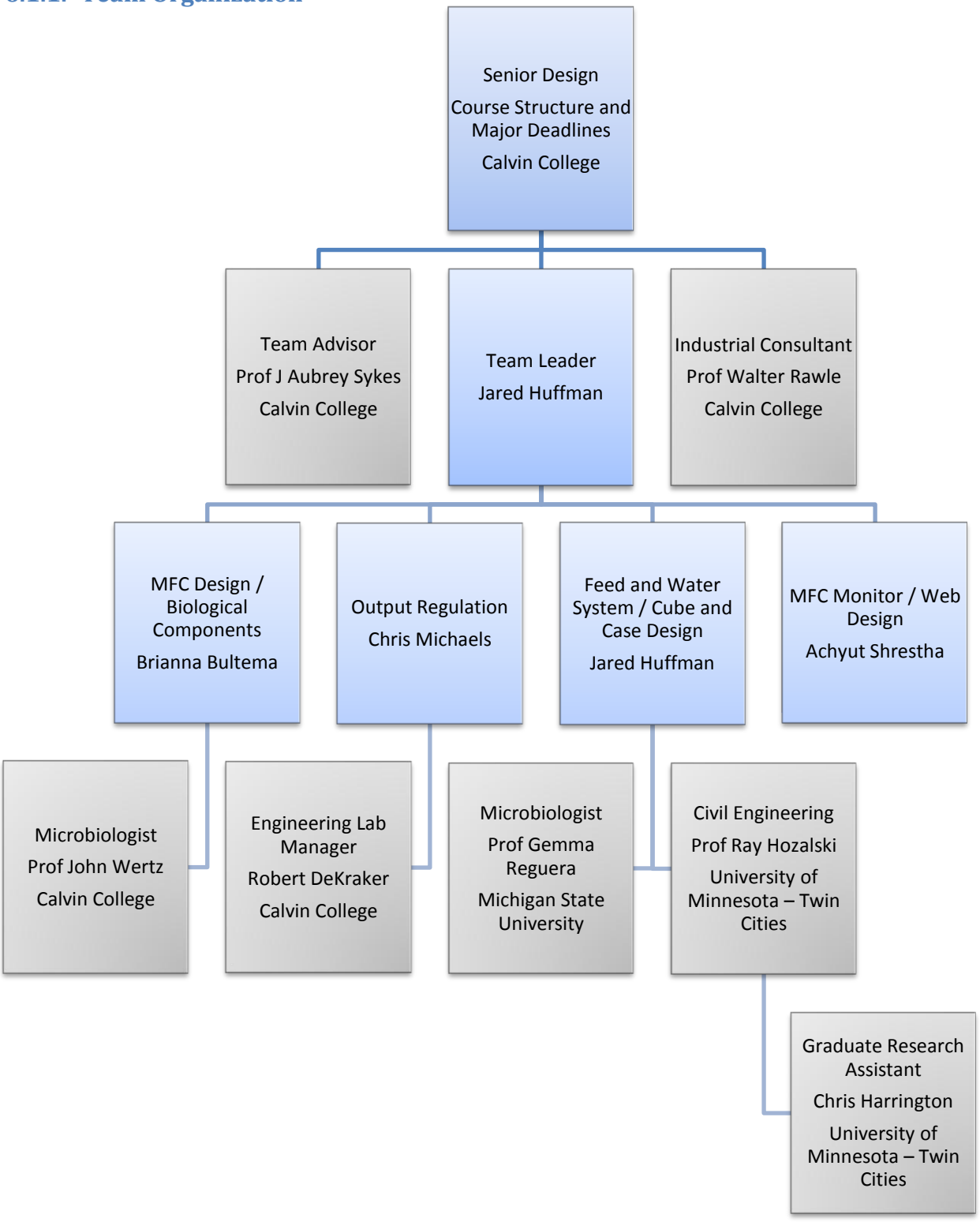



Figure 2 - Team Organization Chart

### 6.1.2. Team Member Bios

**Brianna Bultema, BSE+CHEM**

Brianna is a fourth-year senior engineering student with a chemical concentration and additional course work in biology. She grew up in the shadow of the Rocky Mountains in Colorado Springs, CO where she learned to appreciate and cherish nature. Her biology course work and independent laboratory knowledge make her an invaluable resource for Ecowatt. She has worked at DC Cook Nuclear Power Plant in Bridgman, MI and for a company that specializes in sensors for space, Blue Line Engineering in Colorado Springs, CO. Her passion for renewable and alternative energy sources has stemmed from her internship in the nuclear field and grown through her research and further study on the Biobattery. She is enjoying seeing her contributions become actualized in a final product or prototype.

**Jared Huffman, BSE+MECH/ELEC, BA-PHIL**

Jared is a fifth-year senior engineering student (BSE) with a split concentration between the electrical and computer concentration and the mechanical concentration at Calvin College. He will also be achieving a BA in Philosophy from Calvin College at the same time. He has held a couple of different internships over the years, with the most recent opportunity being a position at Kellogg Company in Grand Rapids, MI; he has worked full time during the summer months and part time during the school year. His favorite Engineering subjects are Industrial Processes and Technical Writing; his course work in the Philosophy department has enabled him to develop a keen analytical sense as well as creative problem solving skills. He grew up mostly in Lowell, MI in a family of seven with two brothers and two sisters; his passion for Project Management originated with the large family unit and extended into frequent leadership positions in various clubs and projects throughout his high school and college education.

**Christopher Michaels, BSE+EE**

Christopher Michaels is a fifth-year Engineering student with an Electrical and Computer Concentration. This past summer he held an internship at the Holland Board of Public Works. He is from Mackinaw City, Michigan. He wishes to design high fidelity audio equipment after graduation, which reflects his musical interests. Also, he would like to utilize his Japanese language skills.

**Achyut Shrestha, BSE+ELEC**

Achyut Shrestha is a senior engineering student with an electrical concentration. He comes from a small town called Damauli in Nepal, home of the Himalayas. His first name literally means “man of integrity” and he gives his best to portray it. He grew up in family where principles are highly valued. He is currently working as a software programmer in Calvin's IT

department. His affinity to learn new things, both educational and social, makes him very adaptable. He excels on pressure and the engineering program in his school has provided ample opportunities. His passion is in cutting edge technology however he wants to use his engineering knowledge and skill to help people from his country where basic necessity is more important than high tech gadgets.

### 6.1.3. Team Member Roles Description

The project has been evenly divided for the four members of the team. Contribution from each team member is essential for the successful completion of the project. Each team member is responsible for completing their assigned sections as well as helping the other members finish their part. Although the divided section is complete in itself, it is of little use without the integration so the integration of the project will require the expertise from the each member in their assigned sections. The role of each member has been properly defined and designated (Table 3 - Team Member Roles).

Jared Huffman is the leader of the project. He is responsible for overseeing that the project is completed in timely manner. As a designer, his primary responsibility is to design a case for the system and integrate the different parts together. His job is to make the product user friendly to the extent possible.

Brianna Bultema is primarily responsible for the biochemical part of the project. Her job description includes, but not limited to, overseeing welfare of the Geobacter and help setup a proper environment for the microbial fuel cell. Her work is closely related to the biology and chemistry aspect of the project.

Chris Michaels is responsible for designing an output regulator circuit. His work is to devise a circuit that will take the voltage from the MFC and stabilize it before optimizing the voltage to a specified output level. He will also be testing the working of a final prototype.

Achyut Shrestha is responsible for designing MFC system monitor. His job is to design a circuit that will oversee the system status and communicate the information to a user in an appropriate way. He is also responsible for testing the initial experimental setup. He also maintains the website for his team project.

Table 3 - Team Member Roles

| | |
|---|---|
| **Brianna Bultema – Biological Components**<br>Biological Research<br>Bacteria Management<br>MFC inner workings | **Achyut Shrestha – MFC Monitoring**<br>Electrical research<br>MFC monitoring design<br>Testing experimental setup |
| **Chris Michaels – Output Regulation**<br>Electrical research<br>Regulator circuit design<br>Testing prototype | **Jared Huffman - Tank / Case Design**<br>Cube Structure<br>Feeding System<br>Tank and Case Design |

### 6.1.4. Schedule

At the start of the Fall 2007 Semester, Ecowatt met to determine a schedule for the Fall Semester. The schedule was determined based on each member's individual schedule as well as a list of project due dates provided by the Professors of Senior Design. The task list for Senior Design was supplemented with other important tasks the team deemed important and relevant to the schedule. The team estimated how many days or weeks each task was expected to require and assembled a Gantt Chart of the results (Figure 3 - Gantt Chart of Fall Semester Plan).

Throughout the Fall 2007 Semester, Ecowatt met regularly to discuss important matters about the project. The schedule was frequently discussed; the team changed the schedule as appropriate to meet more specific or changing deadlines for Senior Design as well as to compensate for unforeseen scheduling conflicts and lead times for ordering materials.

Towards the end of the Fall 2007 Semester, Ecowatt met to again to analyze the team's adherence to the predetermined schedule. The team updated and discussed an "Actual" Gantt Chart for the Fall 2007 Semester (Figure 4 - Gantt Chart of Fall Semester Actual). The research portions of the project, originally perceived as preceding all other testing and prototyping, continued throughout the entire semester. The lead-time for ordering bacteria was not accounted for and thus required an adjustment to the schedule as well. However, the team stayed on schedule within reason, keeping each of the Senior Design deadlines as well as many of the team's own additional tasks.

For the Spring 2008 Semester, Ecowatt planned a schedule much as it did for the Fall, and the goal was to plan for the rest of the year before the end of the Fall Semester. However, the Spring 2008 Senior Design Schedule was not yet confirmed at that time. Prof Wunder was able to provide the Spring 2007 Senior Design Schedule as a starting point; this was later updated to match the official course schedule. From each individual team member's own

schedule and the Spring 2007 Semester Schedule, the team derived a preliminary schedule for next semester, represented with a Gantt Chart (Figure 5 - Gantt Chart of Spring Semester Plan).

Towards the end of the Spring 2008 Semester, Ecowatt met to again to analyze the team's adherence to the predetermined schedule, which was not as close to the plan as the team had hoped. Output regulation redesigns postponed the ordering of parts until later in the semester, and the Biobattery construction (membrane and electrode assemblies especially) proved more tedious and time-consuming than anticipated. Cost-effective gold wire was also particularly difficult to acquire. All in all, the team met the course established deadlines as well as many of the additional deadlines set up at the end of the Fall 2007 semester – being constantly driven by an ambitious planned Gantt Chart.

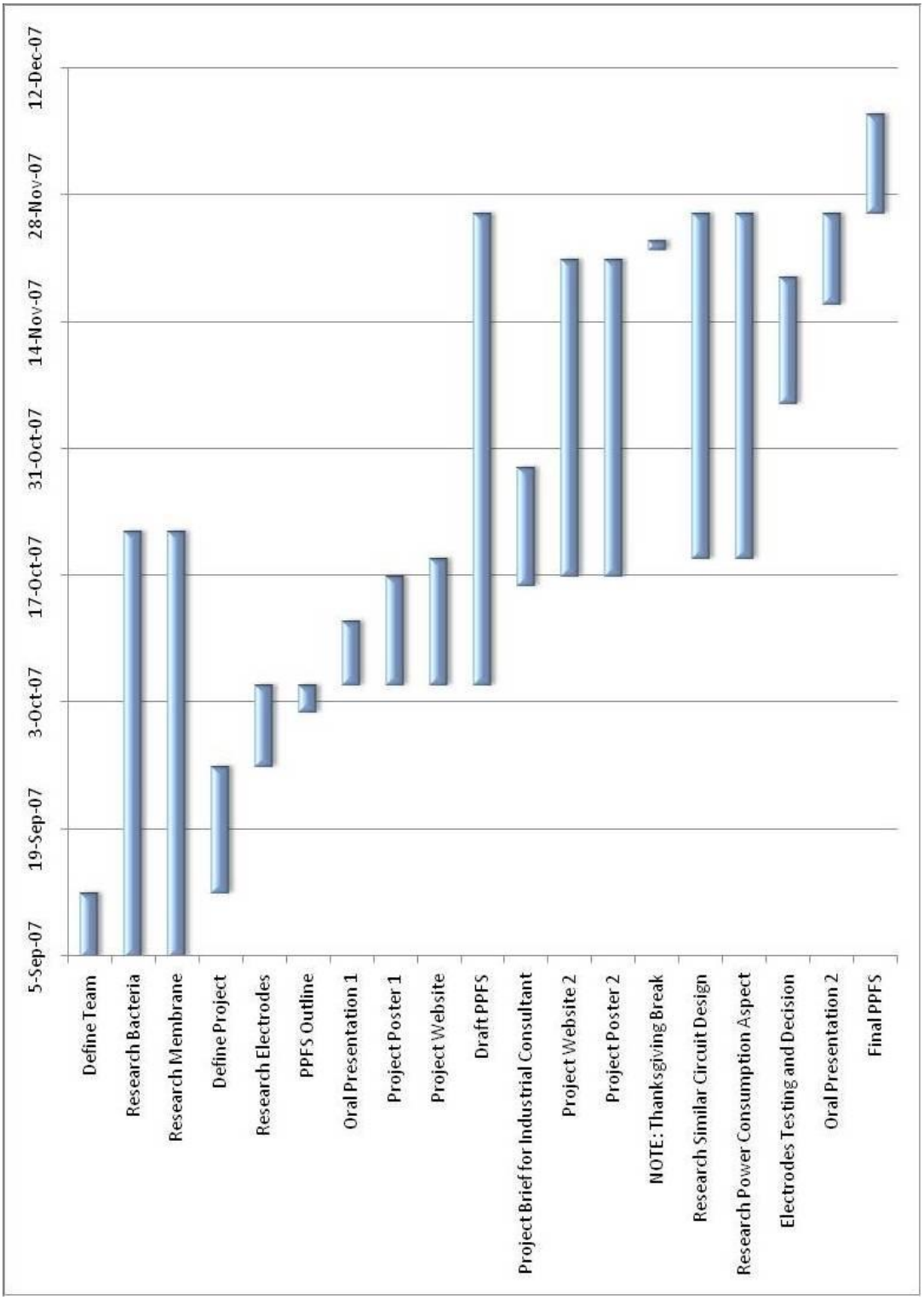



**Figure 3 - Gantt Chart of Fall Semester Plan**

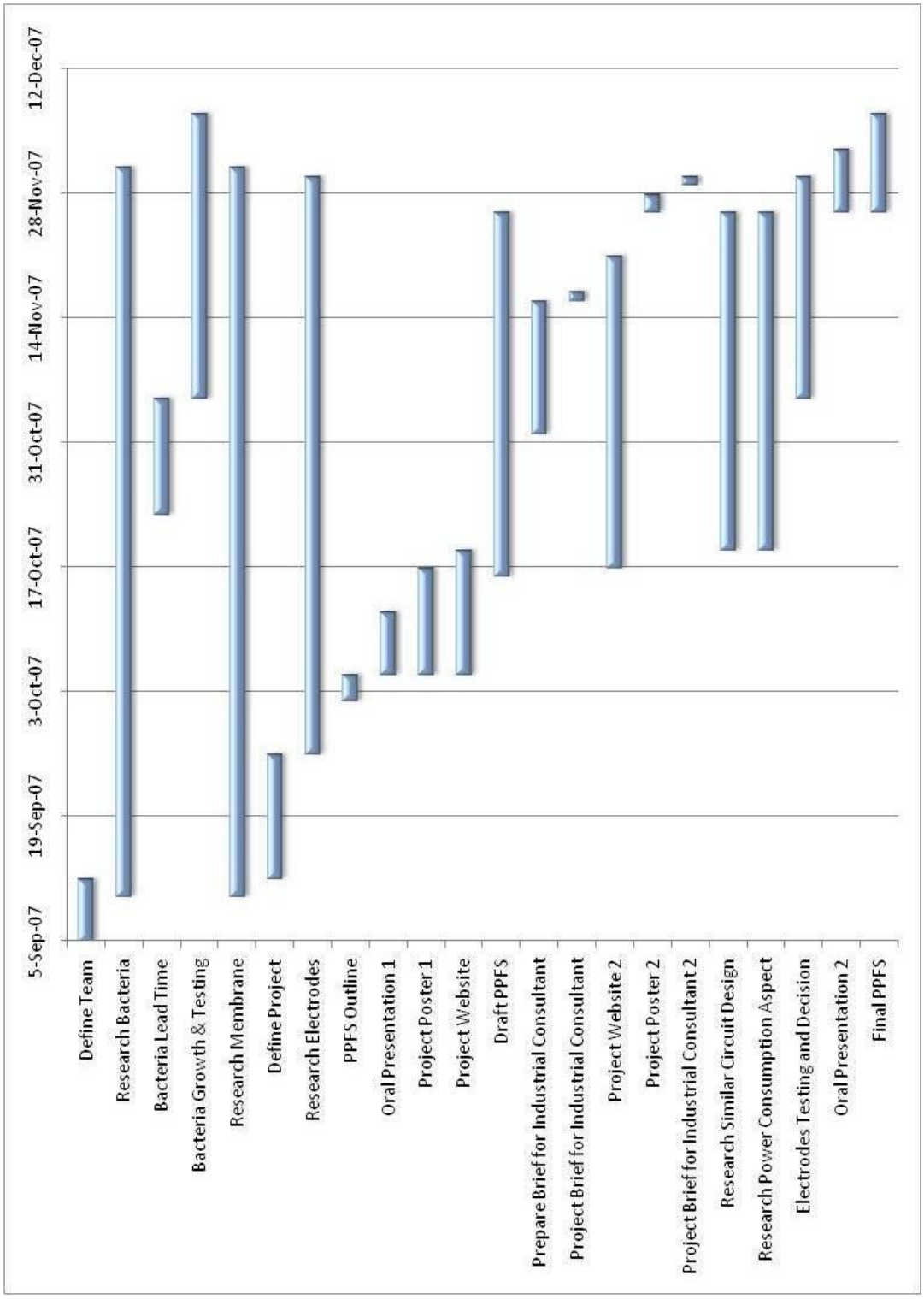



**Figure 4 - Gantt Chart of Fall Semester Actual**

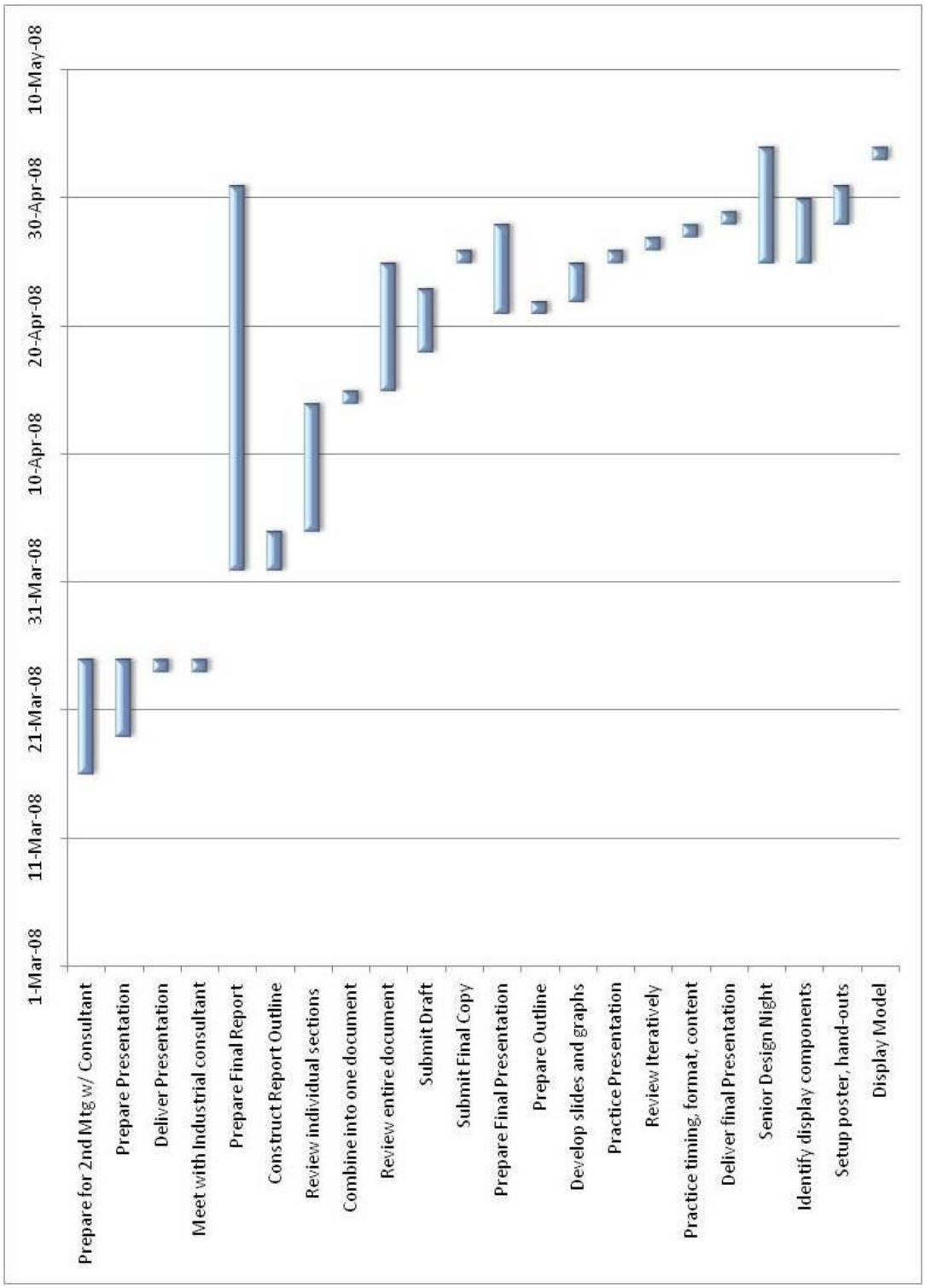



**Figure 5 - Gantt Chart of Spring Semester Plan**

**Project Milestones**

In the beginning of the Fall 2007 semester, there were few milestones identified that would keep on track the scope and the schedule of the project. The time-line for the fall semester schedule was used to identify these milestones. The milestones that would be achieved by the end of this semester were oral presentations, industrial consultant brief, poster design, website design and PPFS report (Table 4 - Fall 2007 Major Milestones and Dates Accomplished). These were the concrete milestones that were given in course requirements. We were able to accomplish all of the mentioned milestones on or before deadline.

**Table 4 - Fall 2007 Major Milestones and Dates Accomplished**

| | |
|---|---|
| Presentation 1 | 10/16/2007 |
| Poster, Website 1 | 10/19/2007 |
| Project Brief for Consultant | 10/29/2007 |
| Poster, Website 2 | 11/21/2007 |
| Final PPFS | 12/10/2007 |

A similar method allowed the team to identify major milestones for the Spring 2008 semester. The team managed to keep to the commitment and complete all of the major milestones on or before the provided deadlines (Table 5 - Spring 2008 Major Milestones and Dates Accomplished).

**Table 5 - Spring 2008 Major Milestones and Dates Accomplished**

| | |
|---|---|
| Poster, Website 3 | 2/18/2008 |
| Project Brief 2 for Consultant | 2/25/2008 |
| Poster, Website 4 | 3/18/2008 |
| Presentation 3 | 3/24/2008 |
| Poster, Website 5 | 4/18/2008 |
| Presentation 4 | 4/28/2008 |
| **Senior Banquet** | 5/3/2008 |
| Website 6 | 5/5/2008 |
| **Final Report** | 5/7/2008 |

In addition to class-provided milestones, soft milestones were created for our specific project. As the Fall 2007 semester progressed, these milestones shaped the project's future direction. These milestones included purchasing Geobacter, creating a working MFC, and testing.

Geobacter, the workhorse of our MFC, was purchased. Getting the right species of bacteria was important because these are expensive and must be handled with great care. After receiving the Geobacter, they are being cultured and reproduced in biology lab. Geobacter will last the life of this project.

One of the defining milestones during the course of the project was being able to successfully test experimental setup of the microbial fuel cell. This was very important because it proved that the project was feasible, at least technically.

The other major milestone was recognizing that the voltage output from the single MFC is too little to power up anything and would have no commercial value. So a multiple MFCs configured in a certain arrangement to generate a standard voltage was proposed and decided. Getting a standard voltage from the cell would make it commercially viable.

Similarly, soft milestones shaped the Spring 2008 semester phase of the project. Identified milestones for the Spring 2008 semester included deciding on the nominal voltage output from the MFC, testing the MFC with membrane, and deciding on layout of MFCs. Acquiring electrodes, gold wire, and suitable membranes also proved time-necessary milestones for the progress of the project as a whole.

### 6.1.5. Contingency Plan for Individual Failure/Absence

As a product, the Biobattery is only viable with all of its features. New products in this field cannot afford superfluous features. Unfortunately, a product with no superfluous features with any unsuccessful or non-functioning parts simply cannot function as a product. Unsuccessful as a product, the Biobattery can still function as a proof of concept demonstration of the functioning features.

**MFC Contingency**

In the event of failure of the MFC aspect of the project, Ecowatt plans to demonstrate the viability of the power regulation system through controlled input voltage. A similar controlled input voltage test will verify the proper operation of the monitoring system design, and the feeding system will demonstrate proper feeding and waste removal on a non-functional dummy MFC.

**Monitoring System Contingency**

In the event of failure of the Monitoring System aspect of the project, Ecowatt plans to demonstrate the viability of the Biobattery MFCs, power regulation system, and feeding and waste removal system. The feeding times and other monitoring functions can be determined with a handheld digital multimeter setup to indicate power status.

**Power Regulation Contingency**

In the event of failure of the Power Regulation aspect of the project, Ecowatt plans to demonstrate the viability of the Biobattery as a power supply with the MFCs, monitoring system, and feeding and waste removal system fully functioning. An unregulated output has limited but undeniable uses, such as powering an LED.

**Feeding and Waste Removal Contingency**

In the event of failure of the Feeding and Waste Removal aspect of the project, Ecowatt plans to demonstrate the viability of the Biobattery MFCs, power regulation system, and monitoring system. The Biobattery prototype can be constructed within a week of the demonstration to show full functionality of the concept. Unfortunately, the power will begin to decrease irreparably after about a week. Each prototype setup would have a limited but fully functioning life.

## 6.2. Cost Estimates for Building Prototype

In making the decisions for what to buy for the Biobattery, Ecowatt met together as a team discuss and debate if the part or parts concerned were worth buying. This discussion had the team member who needed the parts present to the rest of the team what they needed. Then, that team member had to defend why they needed that part and what that part came into play in their design. The rest of the team questioned their decisions critically to make sure that everything done was in the best interests of the team in terms of the product, design, and the budget.

### 6.2.1. Electrical

Due to the fact that Digi-Key, whom Ecowatt order from, has a minimum quantity for the number of parts for any given part that make a part order worth the money, some parts ordered came in more than the needed quantity. However, this allowed for some more error in the process of building the voltage regulator. The parts ordered are in the Bill of Materials in the appendix. For the sake of reference, the parts are here as well: 0.39UH Power SMD inductor from API Delevan Inc., a 12UF 6.3V X5R 1210 ceramic capacitor from Kemet, a 62PF 100V 5% C0G 0603 ceramic capacitor from Murata Electronics North America, a 30V SSOT6 N-channel MOSFET from Fairchild Semiconductor, a Discrete #2 SchmartBoard from SchmartBoard, and a 20V 700MA S-MINI Schottky diode from Panasonic- SSG. For the exact quantity and cost, please refer to the appendix.

### 6.2.2. Case

The cost of the case is largely dependent on undetermined factors. The number of MFCs and their arrangement affects the size and shape of the case. The type of feeding method is also dependent on the MFC arrangement and determines certain aspects of the case design.

The team anticipates the implementation of some type of plastic, but the specific design is not completed. Furthermore, the Refill Station design is not complete. Overall, Ecowatt plans to maintain a Case Cost of approximately $50 with a 100% contingency.

### 6.2.3. Biology Materials

The bacteria, Geobacter metallireducens, ATCC 53774, will cost $175 for a single investment because the bacteria can reproduce without having to buy more. Wolfe's Vitamin Solution, Vitamin Supplement MD-VS and Wolfe's Mineral Solution, Trace Mineral Supplement MD-TMS are valuable solutions for keeping the bacteria alive and healthy. These cost $40/5L of media. The sales order for these materials is SO807459 with the bacteria lot number being 7405983, the vitamin lot number being 57954988, and the mineral lot number being 57954994. These materials came on November 5, 2007, which is 13 days after ordering. The individual battery cost for bacteria will be highly dependent on the final number of MFCs produced.

## 6.3. Business Plan for Full Scale Production of Product

The Biobattery project, as presented, is not quite ready for commercial consideration, so a production-level economic analysis is not currently possible. The future feasible of the Biobattery as a viable product is discussed at length in other sections of the Design Report. The only recommendation the team can make at this time would be to follow the established decision-making processes and core fundamental values of Ecowatt in designing a full-scale production process that provides a good environment for workers, efficiently produces the product, and generates a minimal negative impact on the surrounding local and global environment.

### 6.3.1. Marketing

Because the resulting cost of each unit will go beyond the means of what most consumers will want for a battery, the bio-battery will sell strongest among those concerned about the environment. Therefore, marketing will focus on those willing to pay the price in their cause of good stewardship of the environment. In addition, marketing will show that this device will demonstrate that the bio-battery is a viable alternative fuel to regular lead acid batteries. As such, a push for the scientific community in general to use and research further on bio-batteries is a must.

### 6.3.2. Basic Production Process

A facility interested in manufacturing Biobatteries on full scale level needs to account for certain intricacies of the product. Either the unit must be shipped with the bacteria frozen, or the Biobattery must be fed every 7 days, even during shipment. If the unit is shipped frozen, it will require assembly upon reception. The assembly would be somewhat complex – requiring individual assembly and pressurization of each MFC and then incorporation into the full Biobattery case. Based on this difficulty, the recommended method of shipment is as a fully

functioning, non-frozen unit. At certain stages of the assembly, an MFC is sensitive to other bacteria, so some sterile processes are required. Pouches of the fluid Feed Solution must be included with the product, and refills will need to be sold separately to keep the Biobattery functioning. For more details on Biobattery assembly, see the Biobattery Assembly Instructions section in Appendix I.

### 6.3.3. Budget

The current estimate for the cost of each Biobattery is $200-$500. While the electrical parts will cost less than $35 per unit the rest of the unit will cost substantially more. These costs include the bacteria, the container, the feeding solution, and for the rest of the expenses involved for the actual manufacturing of the battery. The cost of labor and any other costs not listed above are not part of the estimate for the construction of the Biobattery prototype.

## 6.4. Design

The MFC system can be divided into three sub-systems namely the fuel cell, MFC monitor and Output Regulator. The figure below is a basic block diagram of the overall system. The figure below (Figure 6 - System Interaction) shows the interaction between the MFC and the power management systems.

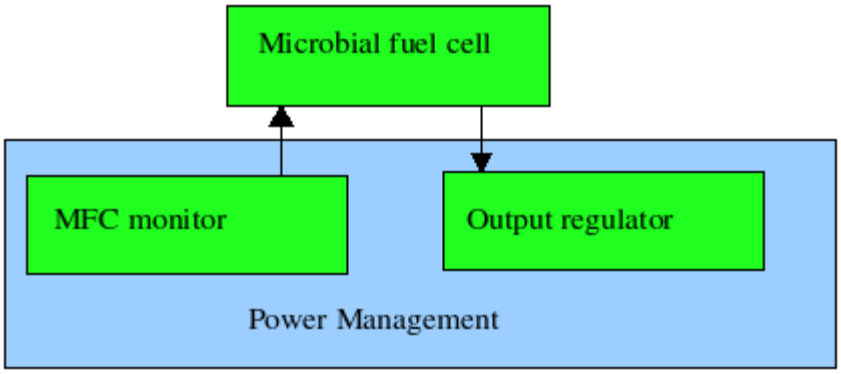



Figure 6 - System Interaction

### 6.4.1. Anode Cube

Current research has not been particularly focused on volumetric power density, but this is a critical factor in the viability of a Biobattery as a realistic power source. In response to this, Ecowatt worked to design a new cell arrangement that drastically improves upon the existing configurations: the Anode Cube.

**Design Considerations / Criteria**

The Anode Cube design had several criteria to meet. The new cell arrangement must easily facilitate the feeding process of many cells simultaneously, but not all cells can share an Anodic Solution due to the electrical combination of voltages. Basically, in order to add voltages, cells must not share an Anodic Solution; they can, however, sum current even while using a shared Anodic Solution. The size of each electrode must be sufficient to provide meaningful power yet small enough to combine in a space-efficient manner, and the final design must be buildable on a prototype and production scale.

**Design Alternatives / Analysis**

Several shapes were considered for the Anodic Structure. Spheres, Cylinders, and Square Prisms were all analyzed as well as Cubes based on surface area to volume ratio as well as ease of construction and food transferability. For all shapes analyzed, an increase in volume causes a decrease in surface area to volume ratio. Furthermore, the cylindrical shapes could be said to most efficiently translate their internal volume into surface area, but the external volume may not be as efficiently used as the more rectangular shapes. Finally, the shapes with rounded parts would be much more difficult to produce for the prototype from available materials.

**Decision**

After analyzing the important aspects of each shape for the Anode Structure, the primary decision-motivator was that of available materials and fabrication methods. Rounded shapes, while producing slightly better surface areas per unit of internal volume, are not easily constructed in a way that provides a sealed oxygen-free environment. Cube shapes, on the other hand, are easily constructed and sealed. Membranes can be attached easily and electrodes can be affixed to the membranes on the cube structure simply enough. Finally, 4" square plates for the sides and top and bottom and 5" square plates for the front and back provide enough space to attach 2" square membrane electrode assemblies without physical contact or difficult to manage corners.

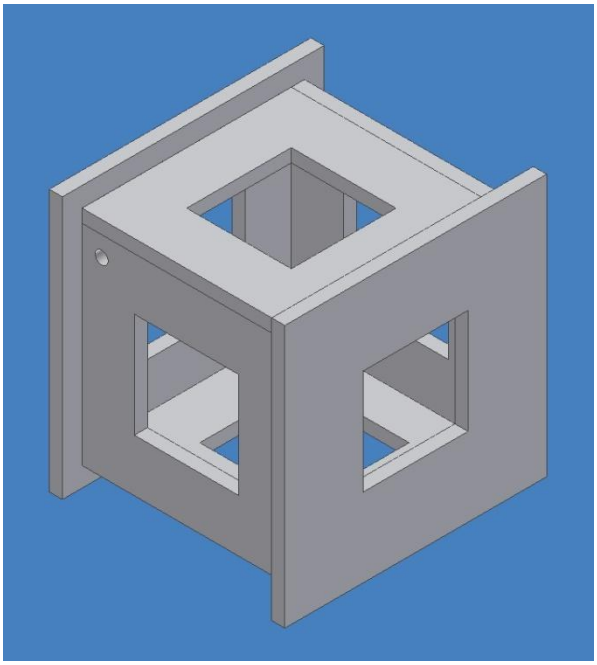


Figure 7 - Anode Cube Model

**Implementation**

Each Membrane Electrode Assembly will be mounted on a 4" X 4" X ¼" Plexiglas plate or

a 5” X 5” X ¼” Plexiglas plate and assembled into a cube structure, known as the Anode Cube. The large plates must be mounted vertically to allow for optimum oxygen diffusion into the Cathode Tank. The electrodes that will have microbes on them will be facing inside the cube, which will contain the Anodic Solution. Many Anode Cubes will be suspended in a single large tank of water to act as a Cathode.

### 6.4.2. Feeding

When the Anode Cubes are first inoculated with Geobacter, they must remain sealed for a period of time to allow a healthy biofilm to form. If this process is not carried out properly, the Geobacter could be carried out of the cubes the flow of food. Once a healthy biofilm is established on the electrodes in the Anode Cubes, a feeding system must be incorporated to maintain healthy Geobacter.

**Design Considerations / Criteria**

A properly designed feeding system will meet the requirements of the bacteria as well as the user. For the bacteria to remain healthy, the feed system must deliver a sufficient, consistent drip of feed solution to each cube. The feed solution must diffuse into the cubes, and it must not contain oxygen or any other harmful contaminants. Finally, the waste must be removed periodically, and it would be counter-productive for the feed system to consume any power.

For the user, the feed system must be easy to maintain. It requires minimal service, and service must be possible without technical training. The waste must be easily disposable, and food should not be wasted.

**Design Alternatives / Analysis**

The typical feeding setup used in research on this type of MFC generally involves large scale feeding systems that consume great amounts of power and are focused on optimum output rather than overall efficiency. For Ecowatt, other options had to be analyzed. Continuous feeding provides the most stable and consistent feed supply for the MFCs, but it is difficult to maintain and generally not possible on the scale of this project. On the other hand, batch feeding is generally too difficult for the average user to manage, and feedings are frequent enough that the process may be cumbersome. To satisfy the requirements, the Biobattery necessitates a hybrid between the two systems types.

**Decision**

The Biobattery incorporates a batch-fed continuous feeding process to provide a steady food supply to the MFCs with minimal upkeep while requiring no additional power. First of all, the Food Supply Bag contains a feed solution in a sealed oxygen-free environment, and the

Water Tank contains water open to the air. The Food Supply Bag is suspended above the Anode Cubes, and a siphoning action draws the food supply from the Food Supply Bag into the Anode Cubes and the waste from the MFCs into the Water Tank below the Anode Cubes.

**Implementation**

The flow of the Food and Water Solutions is controlled by Flow Valves in line with the ¼" tubing to control the feed rate. Additionally, Check Valves are incorporated to prevent backflow issues from the Water Tank to the Anode Cubes and from the Anode Cubes to the Feed Bladder.

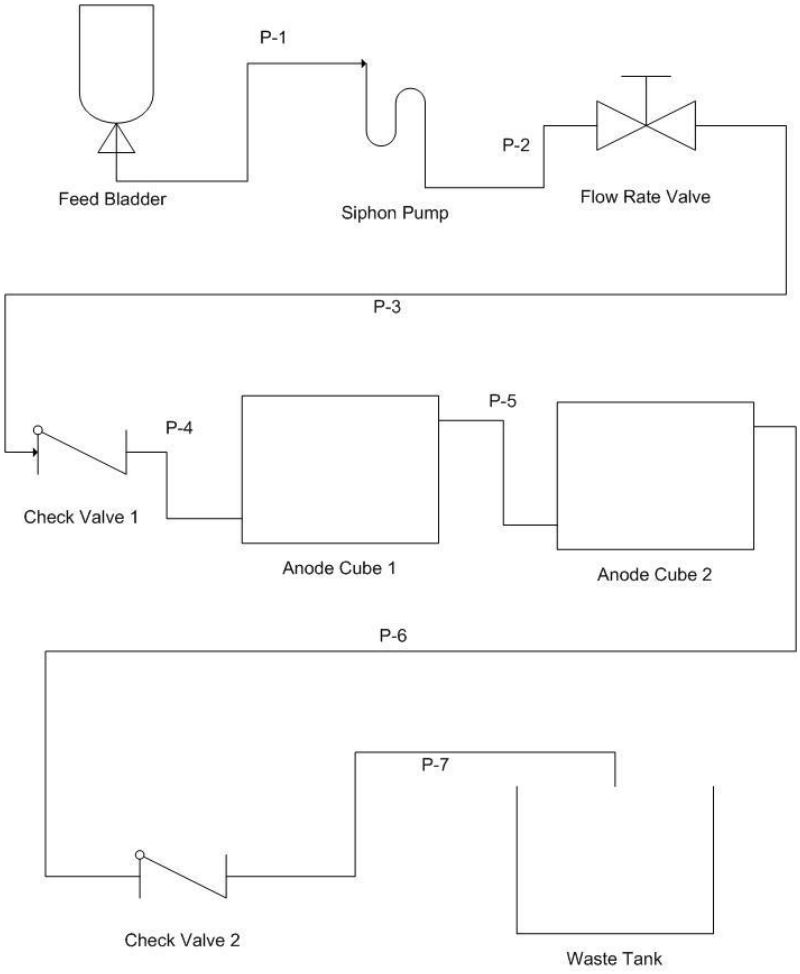



Figure 8 - Feeding System Process Flow Chart

As the Solution Monitoring System indicates on page **Error! Bookmark not defined.**, the user will empty the Water Tank and replenish the Food Supply. The Food Supply Bag must be periodically replaced whenever the bag is empty. The user will return the emptied Food Supply Bag to the manufacturer and install a replacement into the Food Supply Bag Chamber. Reinstalling the Food Supply Bag into the Biobattery case will immediately begin a controlled and continuous feeding process again.

### 6.4.3. Case Design

The Biobattery Case for Ecowatt will house the various solution tanks, bags, electronic circuitry, microbial fuel cells, and the user interface for monitoring the Biobattery.

**Design Considerations / Criteria**

The Biobattery Case must protect the solutions from contamination and leaking, and it should be aesthetically appealing as well as easy to carry. The complete Biobattery should be

lightweight. The case should be made out of a low cost, low density, and sufficiently strong material while being as recyclable as possible; it should fit into a one meter cube.

For the Feeding System to work, the Feed Bladders must be stored above the Cathode Tank, and the Water Tank must be below the Cathode Tank. The Feed Bladders need to be stored in an accessible location, and the Water Tank needs to be accessible for emptying.

**Design Alternatives / Analysis**

Many materials were considered for the case design, but the specifications are very specific. Metal is ideal for recyclability, but the density is much higher than alternative plastics. Wood is attractive, strong, cheap, and available, but once again the weight is a critical factor. Also, wood is not ideal for storing water solutions.

**Decision**

The Case will be made almost entirely out of clear polycarbonate (PC). On the front of the Cathode Tank will be mounted a panel containing the Monitoring System LCD display and concealing the circuitry. PC has optimum strength to weight ratio for holding a large amount of water while not adding significantly to the weight of the Biobattery. It is easily manufactured as clear, allowing for a passive monitoring of the system by the user. Basically, the user can see when the feed bladders are empty and when the water tank is nearly full.

The Case Design is a modular one, with as many similar parts as possible. The design follows a stacked form factor with consistent length and width in stacked trays. The Water Tank and Feed Bladder Tray are actually identical parts which improves recyclability as well as easier assembly and disassembly. The Anode Cubes are arranged in an array that is only a single row deep vertically because otherwise the lower cubes would likely not receive sufficient oxygen diffusion into the water.

The Biobattery with enough cubes to power a USB port under current efficiencies and design fits roughly into a 2’ X 2’ X 3’ cubic space and weighs about 500lbs when filled with water. Further areas to improve on in the future would be efficiency of the electrodes, implementation of Platonized electrodes to eliminate the need for a water cathode chamber (and about 55% of the weight), and research into more effective fabrication methods to prevent leaks and allow for better utilization of cube surface area. Specifically, If the cubes could be constructed by professionals, the plates could be reduced to about 2.5” on a side rather than 4” on a side, drastically reducing the overall weight.

**Implementation**

The following diagrams show the arrangement and design of the Biobattery Case.

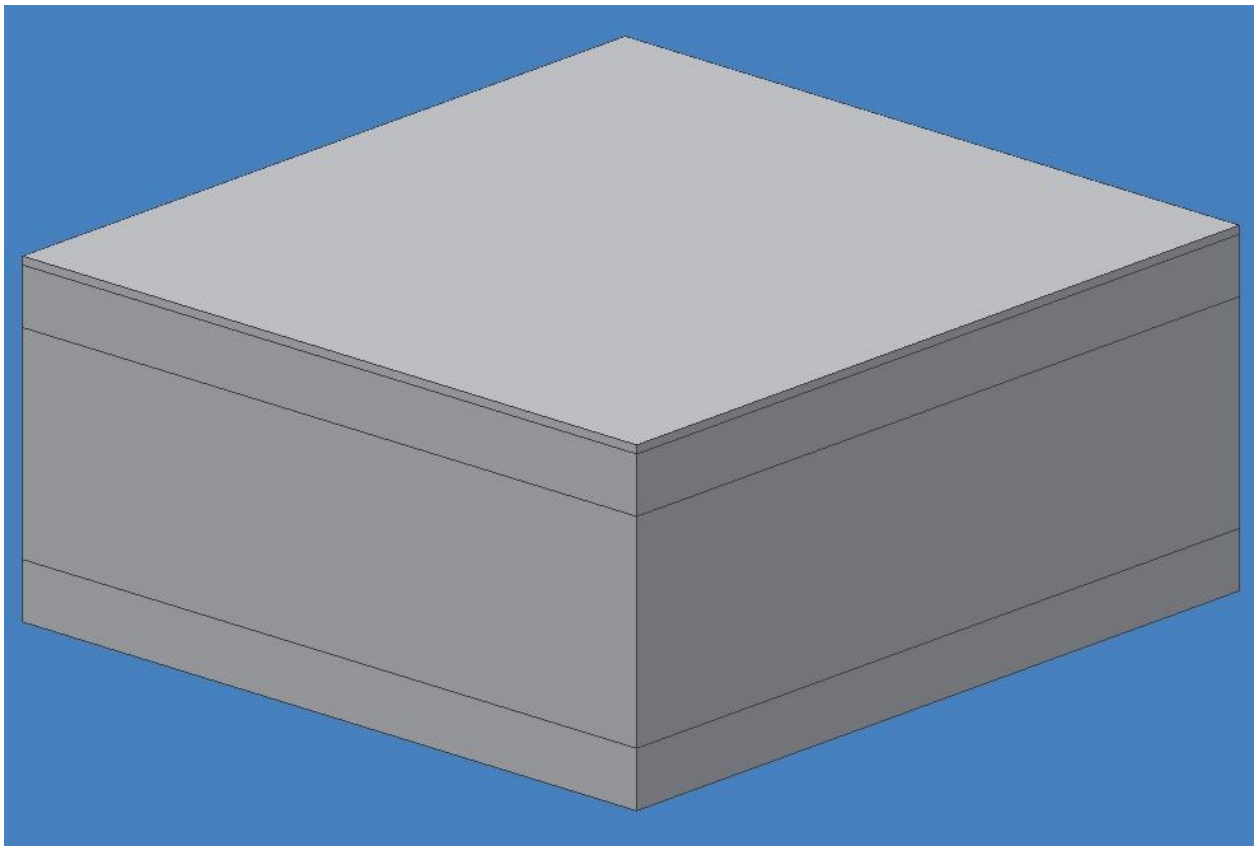


Figure 9 - Full Case - Closed

Table 6 - Case Design Bill of Materials

| Item | Part Number | QTY |
|---|---|---|
| 1 | Anode Cube | 16 |
| 1.1 | Small Face | 4 |
| 1.2 | Big Face | 2 |
| 2 | Support Rail | 8 |
| 3 | Cathode Case | 1 |
| 4 | Food Bladder Case | 2 |
| 5 | Lid | 1 |
| 6 | Food Bladder | 8 |

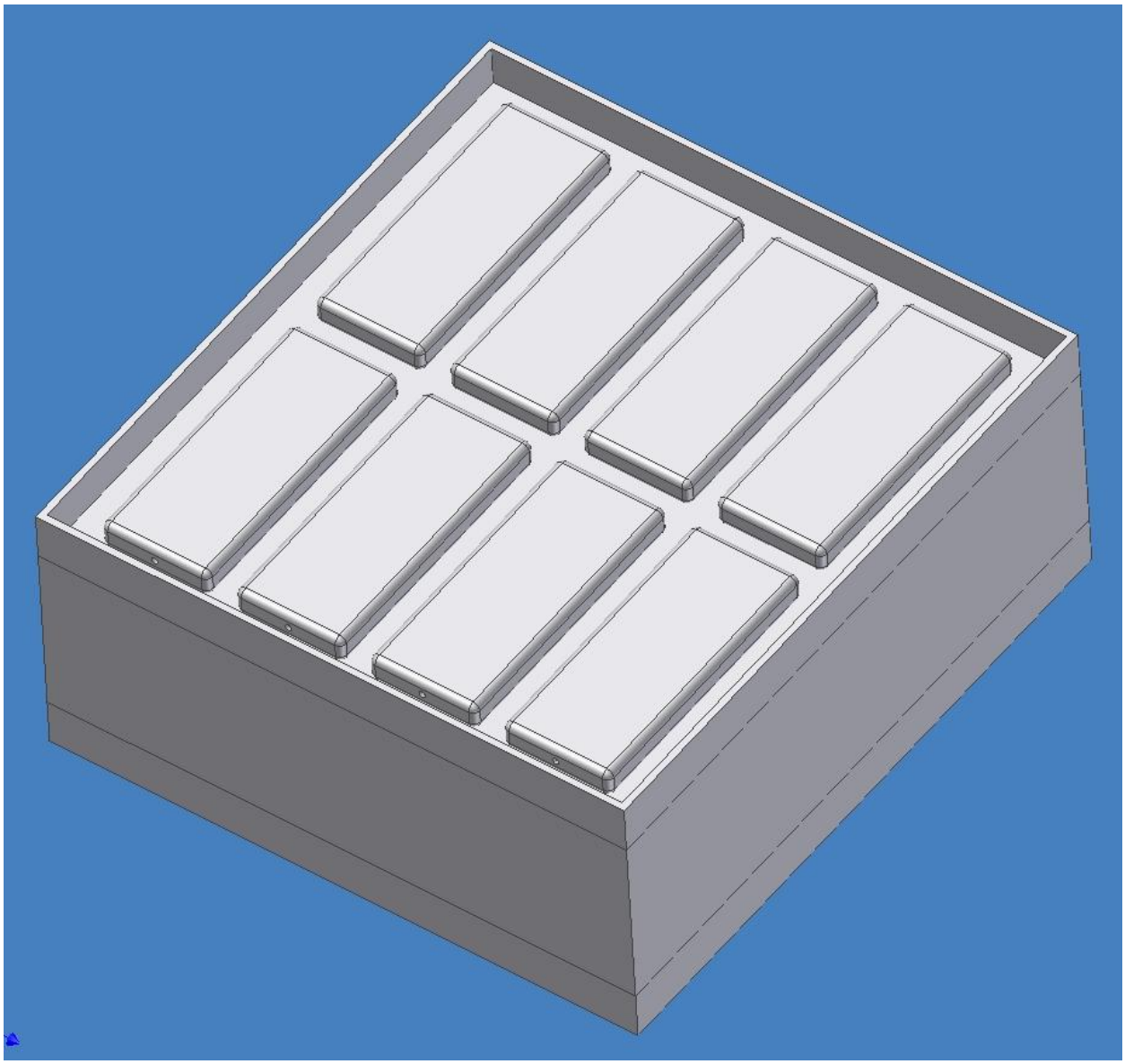


Figure 10 - Case with Food Bladders

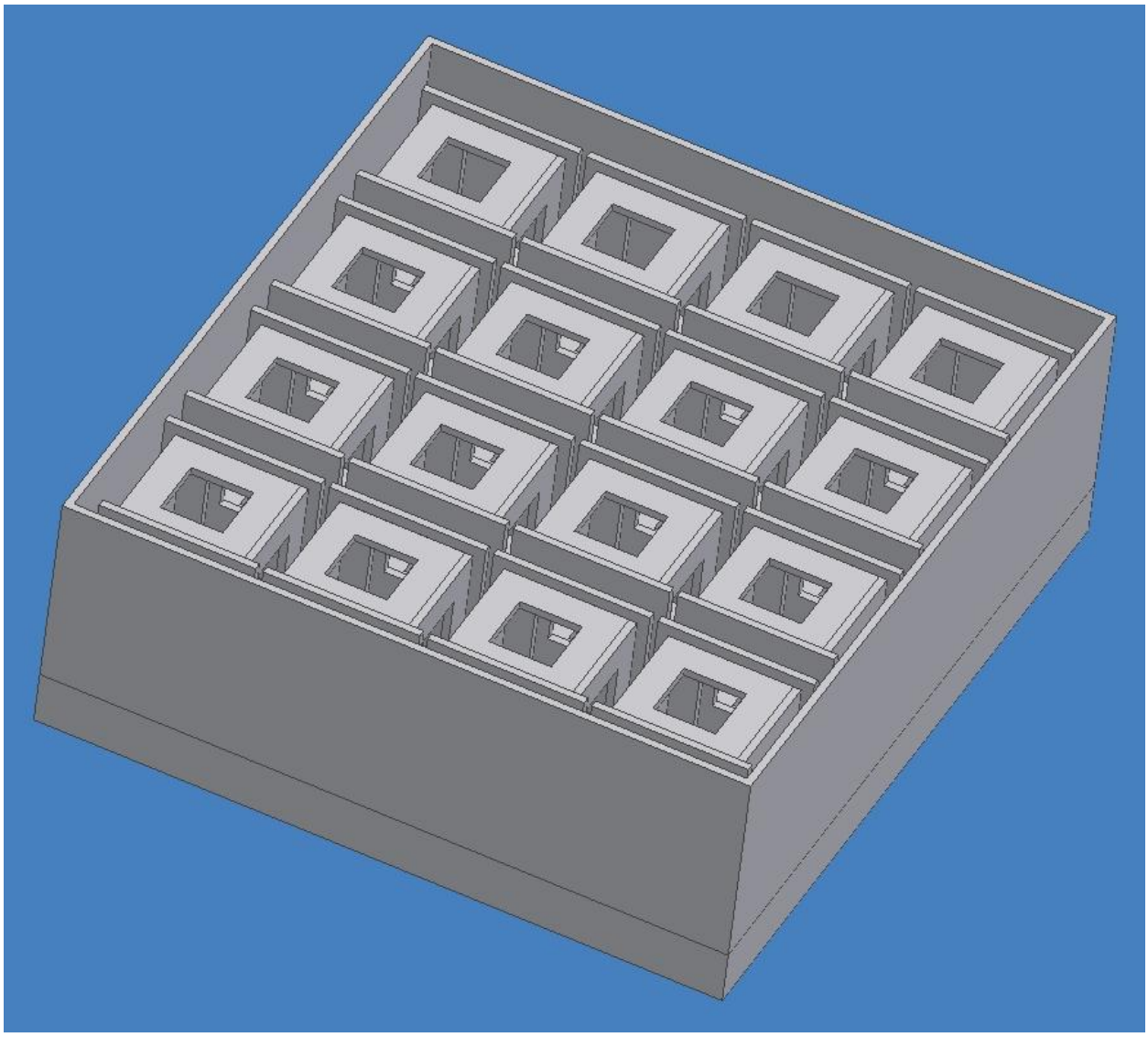


Figure 11 - Case with Anodes Cubes

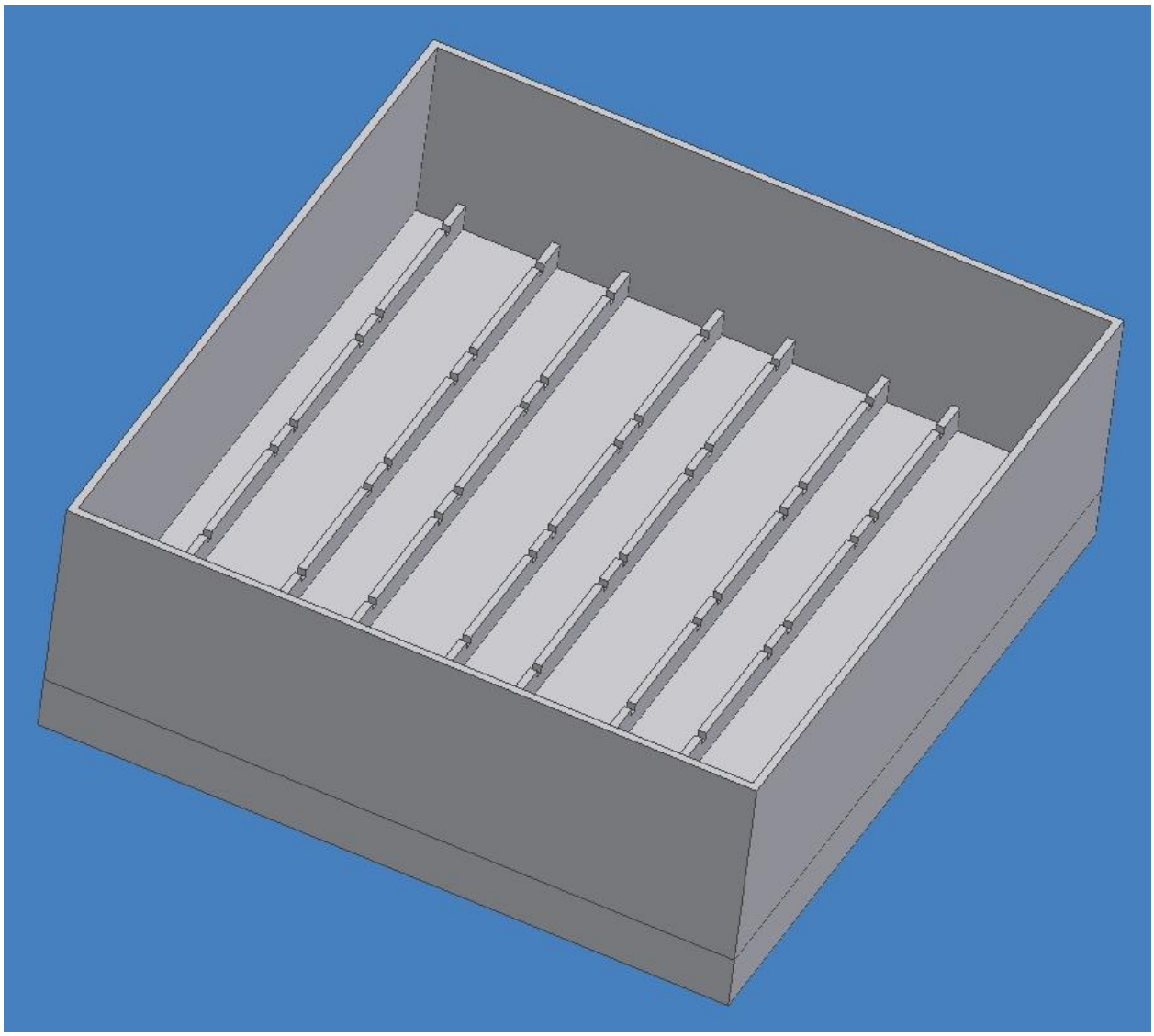


Figure 12 - Case with Support Rails

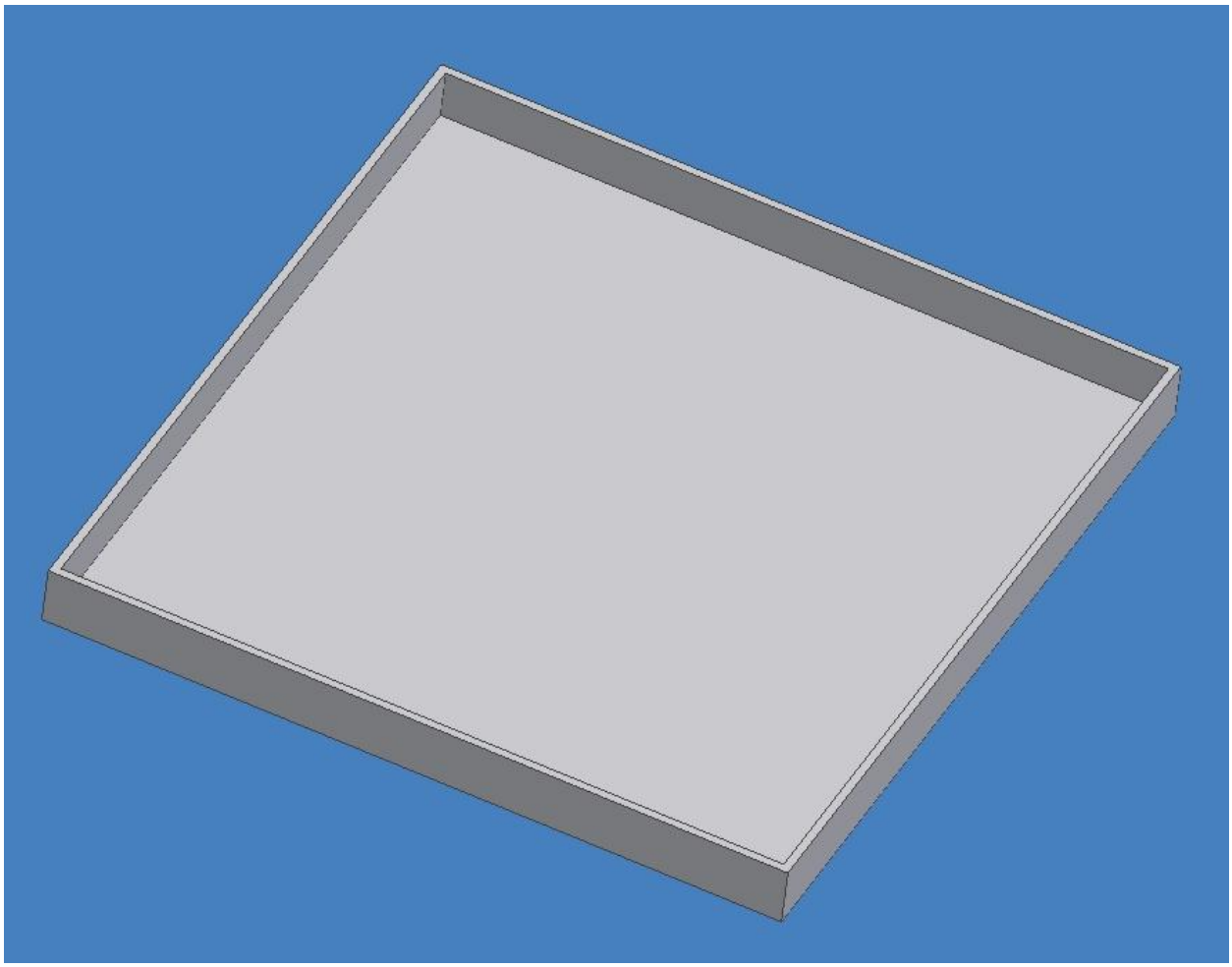


**Figure 13 - Component: Waste Chamber (and Feed Bladder Chamber)**

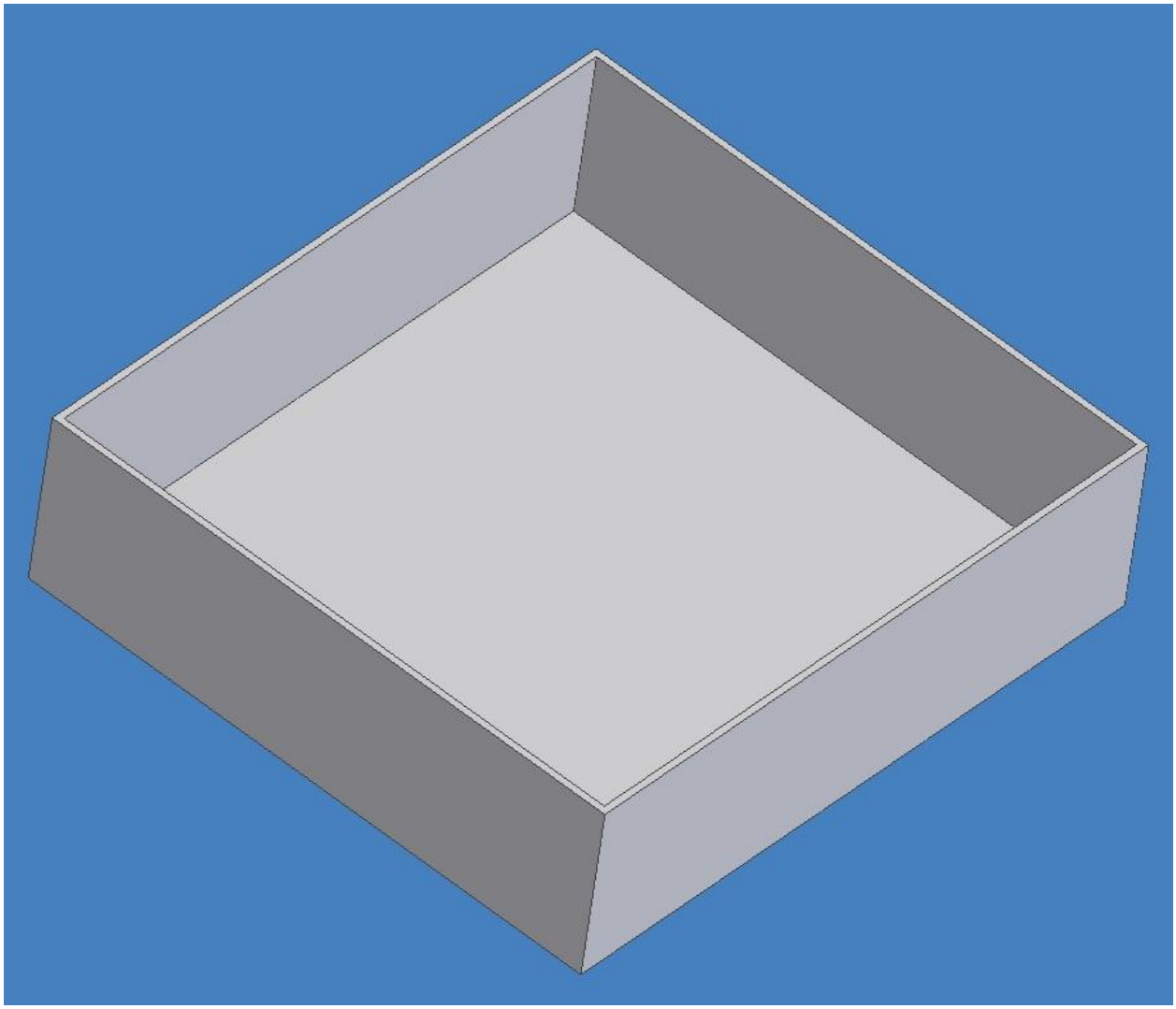


**Figure 14 - Component: Cathode Body**

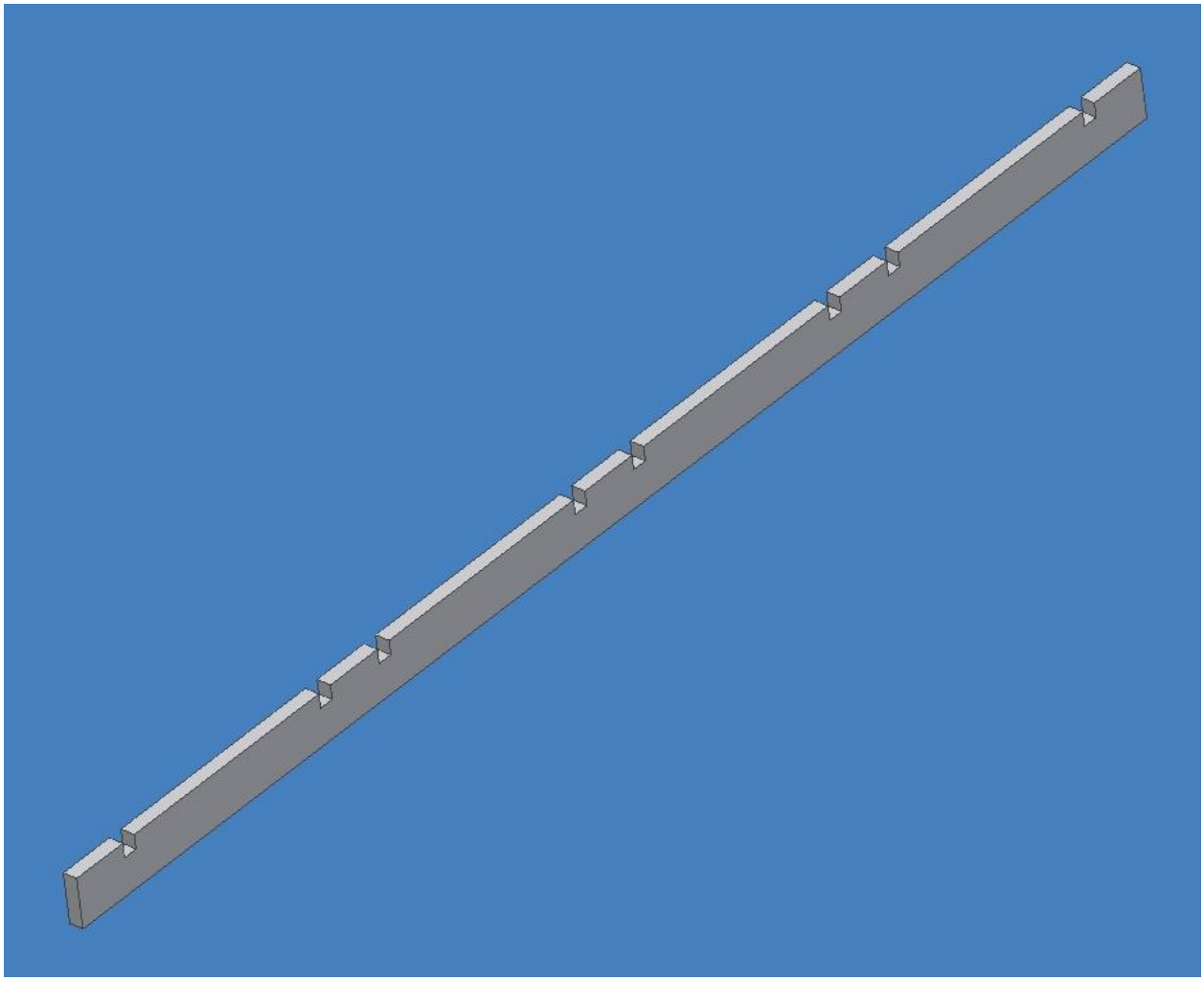


Figure 15 - Component: Support Rail

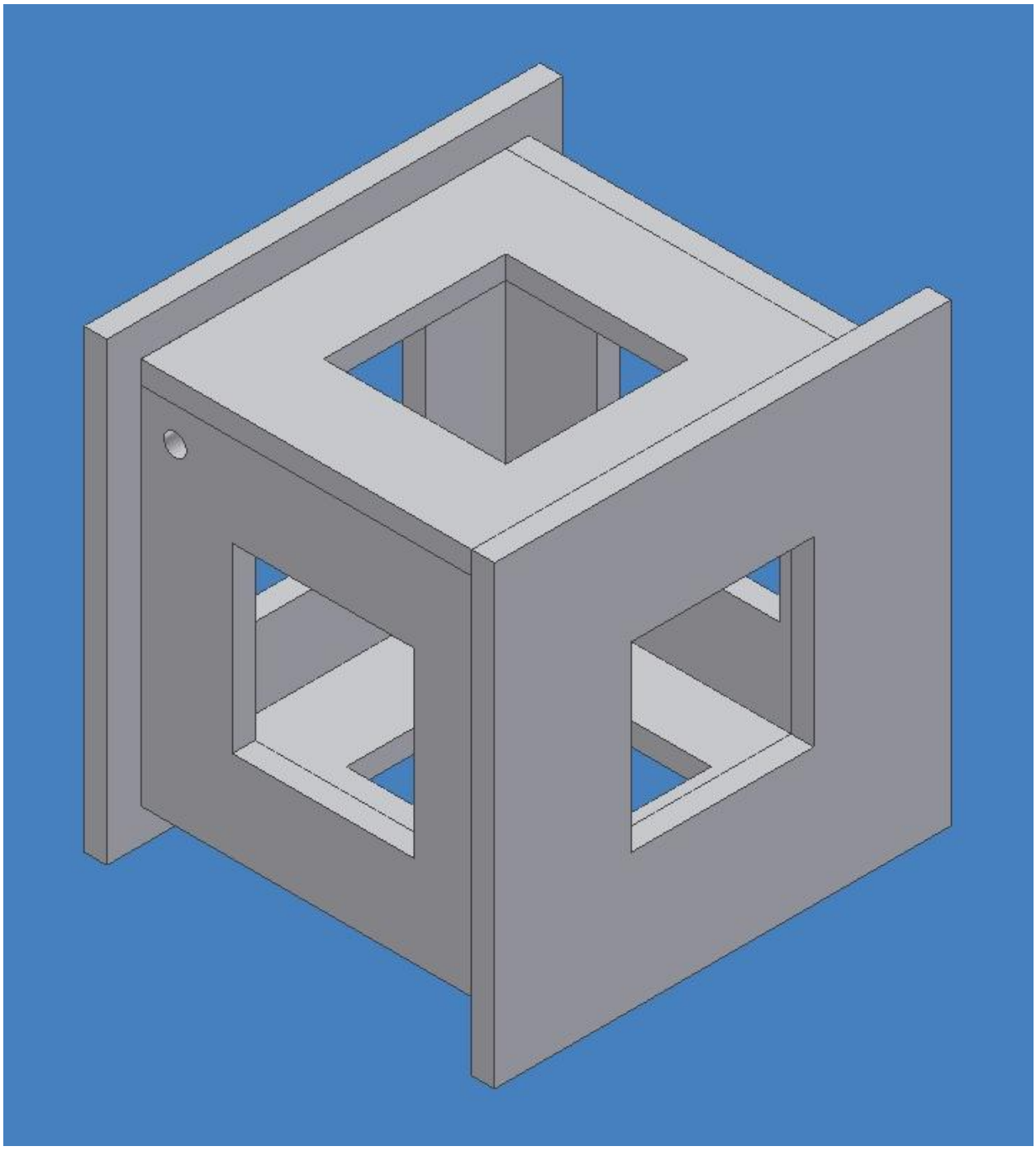


Figure 16 - Component: Anode Cube

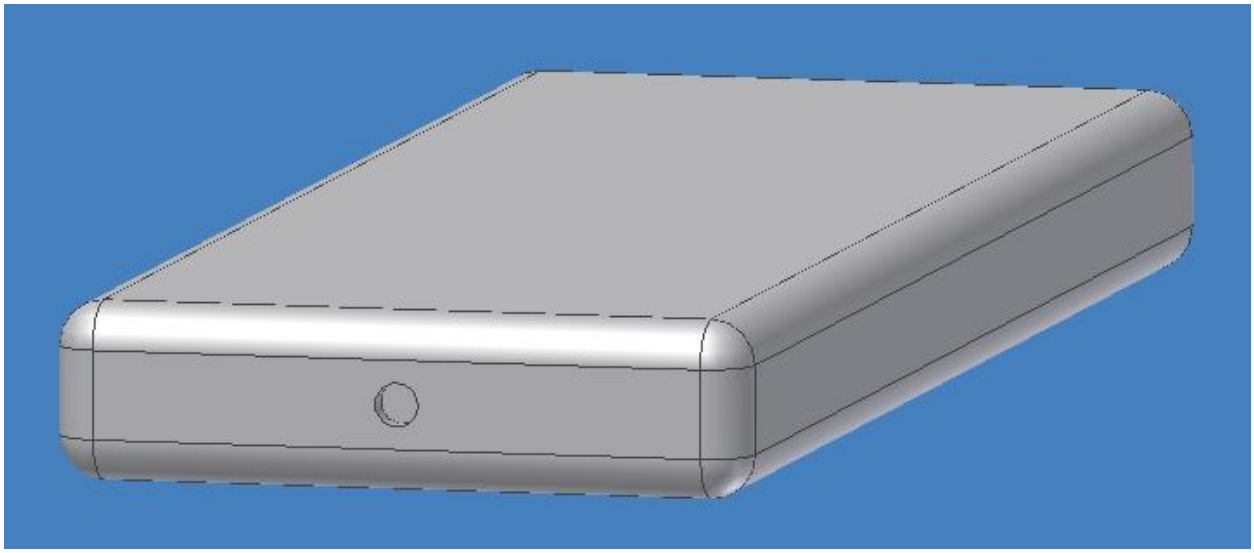


Figure 17 - Component: Feed Bladder

### 6.4.4. Microbial Fuel Cells

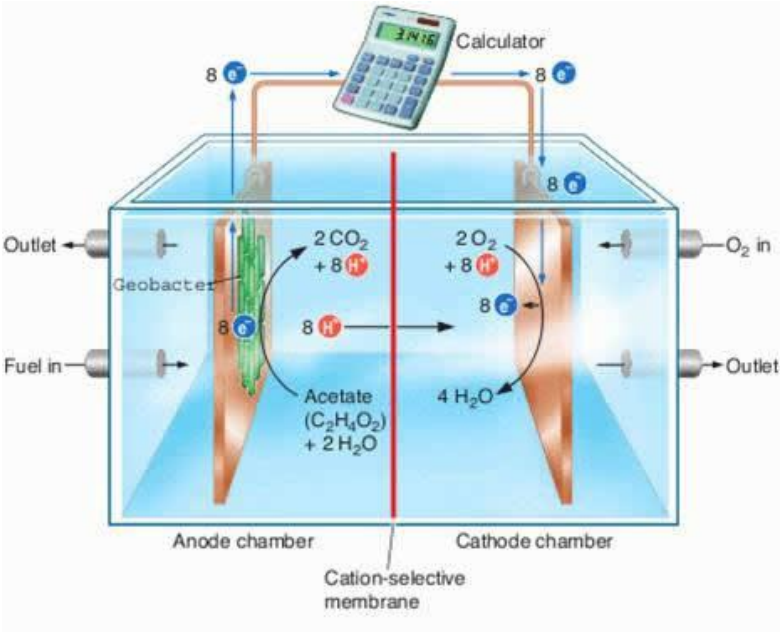



Figure 18 - Schematic of Microbial Fuel Cell

A microbial fuel cell (Figure 18 - Schematic of Microbial Fuel Cell), Derek R. Lovely, 2006) uses the electro-chemical potential difference between a donor bacterium and the final acceptor free atmospheric oxygen. The bacterium gains metabolic energy by transferring electrons from an electron donor, which could be any organic molecule, to an electron donor. A bacterium, in a microbial fuel cell, does not directly transfer the produced electrons to the terminal electron acceptor but are re-directed to an electrode. The electrode that is in contact with the bacterium is the anode. The electrons are then conducted over a wire and are directly converted into electrical energy. (Rao et Al., 1976) The electrons finish at the cathode where they are combined with protons that travel through the proton membrane and oxygen in the solution to form water.

**Bacteria**

Many different possibilities for electron generation from biological species and sources have been developed. Bacteria that can produce free electrons on the outside of their cell membrane are called exoelectrogenic bacteria. The two main groups researched are mixed and pure cultures for microbial fuel cells.

The term mixed culture demotes that the culture of bacteria has more than one species. The mixed cultures composition (ratio of one species to another) changes with each generation depending on what species survives the best in the current environment. Mixed cultures are typically found in nature usually from streambeds and sediment. The sediment battery's generation of electricity is due to mixed cultures of bacteria found in the sediment.

One of the problems with using mixed cultures in a microbial fuel cell is that a symbiotic relationship has been shown to form in certain examples, where one species creates free electrons and one species captures the energy from the free electron. This relationship is not conducive to electricity generation in a microbial fuel cell because the more free electrons in the media then consumed by the second species of bacteria.

Another problem with mixed cultures is that attaining a culture that is efficient in free electron production is challenging to achieve. Many mixed cultures only have a small portion of bacteria that are free electron generators, with the majority simply consuming the organic feed in the media. There have been some cases in the research where mixed cultures have produced a higher power density (Fabaey 2004). One of these cases, using glucose as a carbon source, yielded a power density of 3.6 W $m^{-2}$ (Rabaey 2003).

Isolated pure cultures are more readily analyzable and reliably attainable. The pure cultures refer to the cultures (or growth of bacteria) having only one species present. These pure cultures have been gained by selectively growing a certain species from a mixed culture and testing only that one species. Many environments support these free electron producing bacteria. In fresh water environments, the bacteria *Geothrix* contributes to electricity production. *Desulfobulbus* produces free electrons in the presence of sulfur. *Rhodospirillum rubrum* generates electricity when exposed to ultraviolet light in an anaerobic environment. Many iron reducing bacteria have been discovered. These include "*Geobacter hydrogenophilus,*" *Geobacter sulfurreducens, Geobacter chapelleii*, *Geobacter Metallireducens,* and others. This species of free electron producing bacteria are commonly referred to as the *Geobacter*. *Geobacter* was first described by Lovely et, al. on November 19, 1987. *Geobacter Metallireducens* was the first bacterium that was discovered to completely oxidize multi-carbon organic compounds with the reduction of iron (Lovely 1987).

The bacterium *Geobacter Metallireducens* was chosen from the other possible species explained in the research section because of three reasons. *Geobacter* has the largest power generation to electrode surface area of all researched species. *Geobacter* also is one of the most established and proven species for consistent power generation. It is also obtainable in a pure form from the ATCC.

American Type Culture Collection (ATCC) is a non-profit organization that collects, preserves, and distributes pure cultures of living organisms. The ATCC provides a specific medium for each organism where they guarantee the continued survival of the organism. The medium is ATCC #1768 Broth: Medium for *Geobacter metallireducens*. This ATCC medium production procedure is located in the Appendix IV. The species is anaerobic and produce more electricity when in an anaerobic environment. The gas environment is highly recommended to

be 80% Nitrogen and 20% Carbon Dioxide. *Geobacter metallireducens* also requires Wolf's Vitamin and Mineral solutions.

The feed solution was decided to be the media recommended for quickest growth and reproduction of *Geobacter*. This increased activity corresponds directly with the consumption of acetate and production of free energized electrons. For setup of microbial fuel cells are found in the appendix. The typical medium used for the *Geobacter* species includes ferric citrate as the electron acceptor and sodium acetate as the general organic feed. These chemicals are, for the purpose of this project, obtained from the biology department at Calvin College.

**Electrodes**

The MFC will include two electrodes, anode and cathode, for transferring electrons. The anode is the negative terminal where the electrons are deposited and transferred by Geobacter as electrons gets generated. The cathode is the positive terminal where the electrons will ultimately be transferred from the anode. The buildup of electrons in anode causes the potential difference between the two electrodes so the electrons flow from anode to cathode generating electricity.

The two electrodes are made from the same material. Copper electrodes are unsuitable for this design since copper has detrimental effect upon Geobacter. Any non corrosive metal will work for the use of electrodes. Porous graphite electrodes will be used for this particular design. Electrodes can be made in different shapes and sizes. For the use of this design, electrodes will have a bar shape. The potential of the electrode increases with the surface area of the electrodes so a porous electrode is preferred since it has larger surface area. With a greater number of pores, there is a larger surface area available for electron collection. The greater number of pores also helps prevent clogging. In the anode compartment, Geobacter transfer the electrons to the electrode. Pili, also known as nanowires, are the appendages of Geobacter, the electrons produced are carried to the electrode via these pili as these Geobacter cling on the surface of the electrodes. The cathode, which is also the graphite electrode, collects the electrons which flow from the anode. In the cathode compartment, oxygen acts as the final electrons acceptor. The chemical reactions that take place in the cathode compartment are written below.

$$O_2 + 2e^- \rightarrow 2O^- \qquad \text{(Eq. 1)}$$

$$2O^- + 4H^+ \rightarrow 2H_2O \qquad \text{(Eq. 2)}$$

Oxygen is suitable since it has high oxidation potential, it is freely available and it combines with hydrogen ions coming via membrane to form water, so there is no harmful waste. The original design used a water tank with the electrode submerged in water. To reduce the weight of each MFC, an air cathode is proposed. This cathode must be plated with a metal that increases the rate of reaction in Eq. 2. These catalyst metals include platinum coated carbon electrodes, and palladium electrodes.

Carbon cloth electrodes were donated by University of Minnesota, Twin Cities, Civil Engineering Department. These cloth electrodes provide a large surface area for *Geobacter* to populate and are flexible and thin to reduce the diffusion distance from electrode to membrane to electrode.

**Proton Exchange Membrane**

From the research conducted and recorded in the papers shown in the research section, two proton exchange membrane options were presented. The first was a Nafion 117 membrane that is traditionally used in research Hydrogen Fuel Cells. This membrane is thicker and more robust than the alternative of natural cellophane. The Nafion membrane was discovered in 1960 by creation of sulfonated tetrafluorethylene which was permeable to cations. Nafion is resistant to chemical attack and impermeable to water. The Nafion information sheet is located in Appendix VI. The main drawback to the Nafion membrane is the

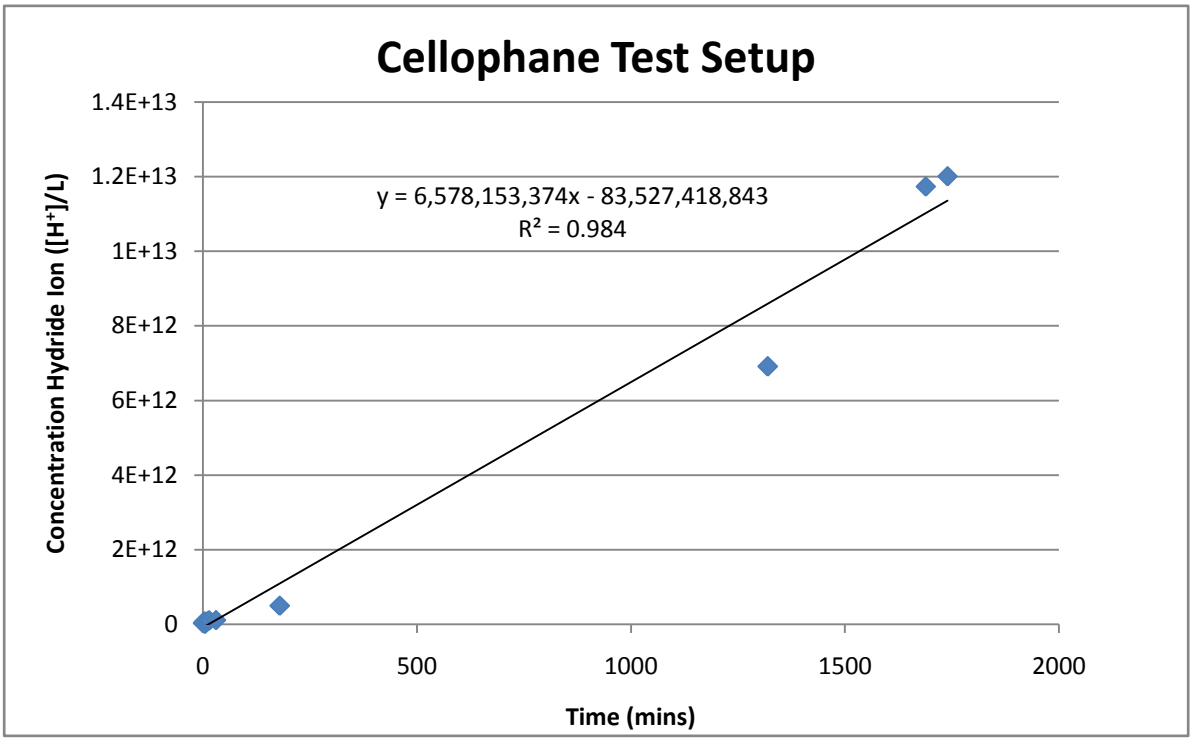



Figure 19 - Cellophane Membrane Testing

high cost. Nafion, with appropriate thickness for Ecowatt, would cost $297 plus shipping and handling for an 8" by 10" rectangle. Ecowatt had the opportunity for a fellow researcher, Professor Ray Hozalski at the University of Minnesota, to provide a sample of Nafion 117 for testing.

Natural cellophane has similar proton exchange properties, not as well researched, but less than $5 for more than 5 yards. Testing was completed to compare the proton flux through a 1cm circle of membrane. Testing Data may be found in Appendix V. The Cellophane was found to have a proton flux of 85% of the more expensive Nafion membrane. This 15% decrease in proton flux does not justify the use of the significantly more expensive membrane. The cellophane membrane was the chosen membrane for final design.

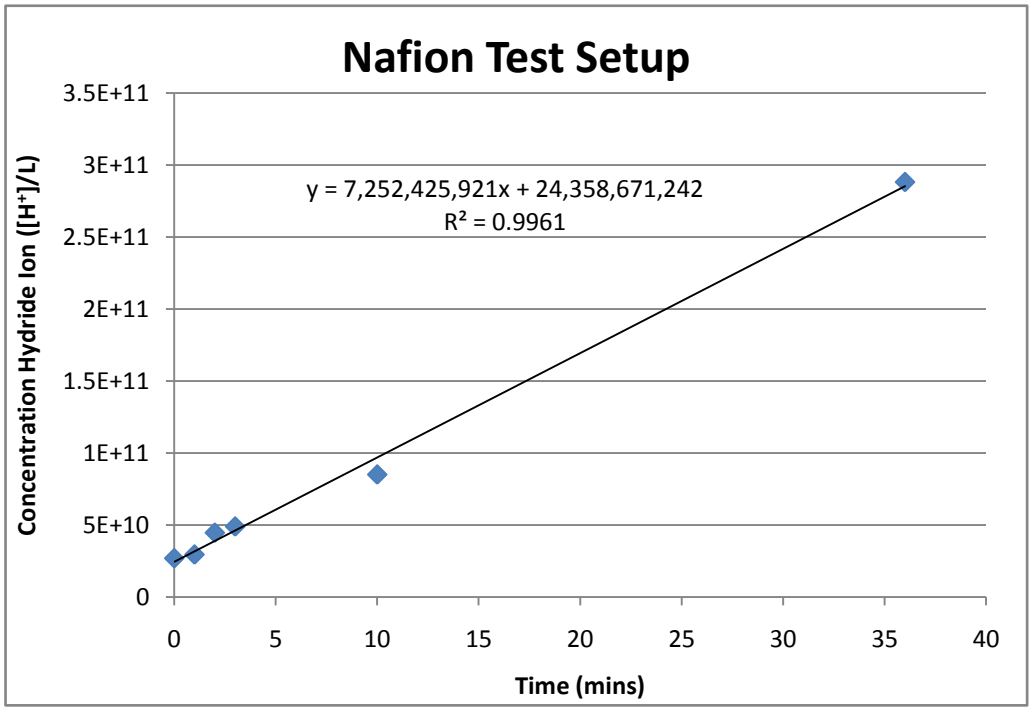



Figure 20 - Nafion117 Membrane Testing

### 6.4.5. Fuel Cell Arrangement

Three options exist for the fuel cell arrangement; series, parallel or both. When the fuel cells are in series, the voltages of each cell will add together. If one of the cell's outputs becomes unusable, it will short the circuit between the rest of the cells. In the final design, this has the potential to become difficult to repair. As a result, one small malfunction can cripple the entire battery, such as the death of the bacteria in one of the cells.

The second option is arranging the fuel cells in parallel. While this option can possibly suffer from lower voltage output, the current is additive. If one cell fails, the same voltage will result.

For the purpose of having a reliable current and voltage, the fuel cell arrangement decision needs careful consideration. Therefore, Ecowatt decided to use a combination of both parallel and series for the fuel cell arrangement. The fuel cell arrangement requires a compromise between the two of these options, which leads to the third option. By placing a small amount of resistance between the fuel cells and running half of the fuel cells in series with the other half in parallel, one can simultaneously receive the result of having a circuit in both series and parallel.

### 6.4.6. Power Management

The next major module in the bio-battery design is Power Management. The voltage produced by the MFC needs regulation before powering a load is feasible. A USB female port will power a load. In order to power a USB device, a USB power switch is necessary. Along with the regulation of voltage, Power Management module will also monitor the system status. The following diagram is a basic inflow and outflow of voltages and signals.

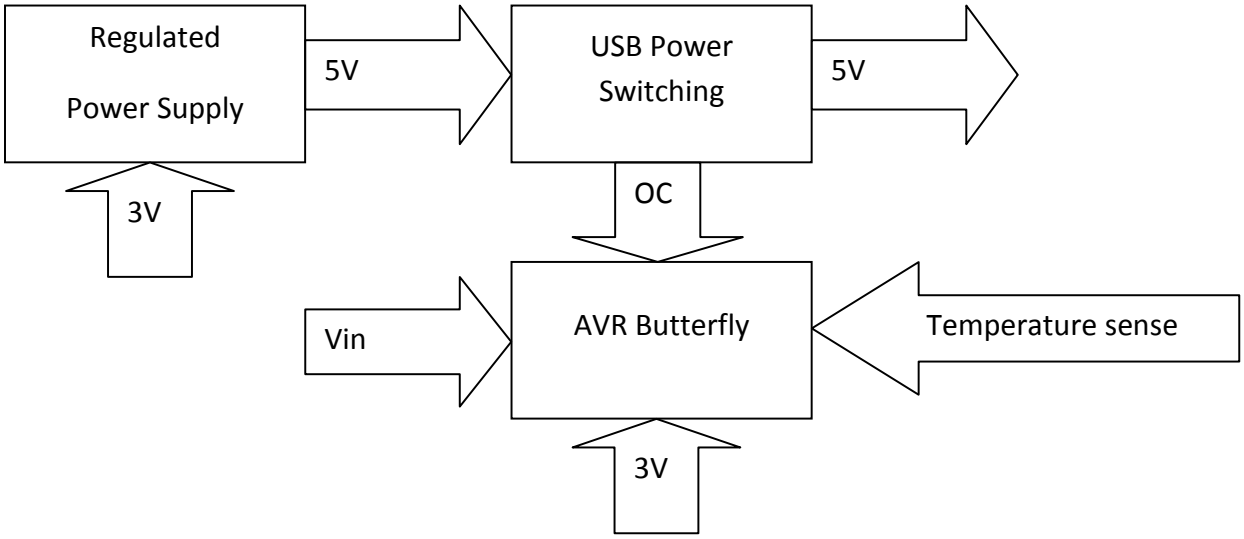



Figure 21 - Power Management Block Diagram

### 6.4.7. Feeding

Microbial fuel cells contain bacteria that must have a sufficient supply of “food” to survive. In fact, the power generation performance of a microbial fuel cell is dependent on food supply. Since the bacteria must live in an anaerobic environment, the method of adding the food to the cell solution requires finesse.

The food supply can be added to the cell solution periodically in controlled amounts through a process called batch feeding. At feeding time, the solution must be maintained while the food supply is added and the waste solution is simultaneously removed. Unfortunately, batch feeding must be done at relatively frequent intervals, and the power output can drop significantly starting about one week after last feeding.

To solve the problems associated with batch feeding, researchers have employed continuous feeding. Continuous feeding maintains a steady flow of food supply into the cell and an equal flow of waste solution out of the microbial fuel cell, requiring little maintenance or upkeep. On the other hand, continuous feeding systems are complicated and expensive to implement. Pumps would be necessary to continually draw the food supply into the microbial fuel cell; the power for such pumps would use more energy than the Biobattery could possibly generate, rendering them unfeasible.

To optimize the benefits and overcome the drawbacks to both batch and continuous feeding processes, a combination of the two methods is implemented in the design. The batch-fed continuous feeding process will be discussed in detail in the Design section.

### 6.4.8. Regulation

Because the circuit requires low power usage, one specialty circuit component, the MAX1524, is important. This part is a Simple SOT23 Boost Controller that will adjust the voltage at a frequency of 296 kHz. The MAX1524 consumes little power while maintaining stable performance and uses bootstrapping to increased performance in chip start up. The MAX1524 also has the ability to detect faults to protect the rest of the circuit from low voltages, where the chip will try to restart itself to maintain optimum performance. While other parts received consideration for usage in the design of the regulator, none of the other parts achieved the desired results.

The other required parts in the circuit are a 3.9UH Power SMD inductor from API Delevan Inc., a 12UF 6.3V X5R 1210 ceramic capacitor from Kemet, a 62PF 100V 5% C0G 0603 ceramic capacitor from Murata Electronics North America, a 30V SSOT6 N-channel MOSFET from Fairchild Semiconductor, a Discrete #2 SchmartBoard from SchmartBoard, and a 20V 700MA S-MINI Schottky diode from Panasonic- SSG. All other resistors and capacitors are available in the Calvin College Electrical Engineering Lab.

The voltage produced by the MFCs is not constant enough for consideration as a reliable power source, so regulation is necessary for application. The voltage regulator adjusts the input voltage to a set internal reference voltage. This means that the voltage regulator is programmable by means of a network of analog devices on the outside of the voltage regulator chip itself. By internally switching the voltage to check the voltage, the MAX1524 is capable of

putting out a desirable voltage. Calculating the correct switching frequency and maximum duty cycle reduces the amount of noise produced by the circuit and Biobattery at the output. The voltage regulator is capable of leveling relatively small changes in fuel cell output.

**Design Criteria**

The typical microbial fuel cell output is 0.5 volts per cell. To achieve a higher voltage, multiple cells are used in series to increase the voltage. For reliability, 3.0V is the set minimum voltage. A minimum of six cells are necessary to keep the battery, the regulating circuit, and the measuring circuit working at optimum performance.

The battery's circuit design yielded four options. Each design carefully considered the needs of the circuit and the needs of the battery. The needs of the circuit included minimum voltage, current, and power to enable the circuit to operate correctly. Beyond this point, the circuit needed adjustment to make sure that the voltage level of 5.0V and current level of 100mA to 500mA (preferably around 300mA) came from the voltage regulator's output. The circuit functionality must be verifiable and not harm the microbes in the MFC.

**Design Alternatives and Analysis**

The first design used specialized parts, such as a low input linear voltage regulator and a low input voltage bipolar junction transistor (BJT). If the current output of the MFC dipped, the resulting voltage will not affect the output voltage and the circuit's functionality. This design has an active voltage measurement that increases the energy loss in the circuit.

The second design used a TPS62100 switching voltage regulator from Texas Instruments in a configuration with a diode, four resistors, and four capacitors. This design requires more power than the second design but this design is 40% more efficient. This circuit requires an additional current source. The problem with this addition is that the current source circuit requires an external reference in order to supply the desired current. Given the scenario with the Biobattery, this is not possible to do, since it would require an addition source of power that the Biobattery could not supply. One of the pins has a high current requirement too to activate the chip into a running mode.

The third design would implement a Maxim Simple SOT23 Boost Controller in conjunction with one inductor, one diode, one MOSFET, three capacitors, and three resistors. The design will take about 25µA of power for the chip to run correctly, but it will have an output of about 85-90% efficiency. Hence, it is even more efficient than the previous design. In addition, this design allows for more flexibility in terms of designing to the needs of the Biobattery. However, the MAX1524 and most of the parts needed for the correct design implementation require surface mount parts, which Team Ecowatt does not have the facilities

to build on a circuit board. Therefore, a SchmartBoard is necessary to solder all of the surface mount parts utilized in the fourth design.

The last design suited the needs of the Biobattery's regulation unit. This design will give low-cost power at the desired power rating. Moreover, the MAX1524's built-in short-circuit and fault protection make it ideal for the variable voltage and load coming from the Biobattery cells and the user's application.

**Simulation**

While simulating the circuit is ideal, this was not possible. In the midst of searching for parts, Ecowatt desired to find a circuit with a simulation file for PSpice. Since having a simulation file was only one of the design criteria for finding a part and more important design criteria existed, Ecowatt found that no part existed that fit both the rest of the design criteria and the criterion for having a simulation file. Even though this result is regrettable, Ecowatt will still simulate the circuit through testing. The circuit went onto the board for soldering without any computer analysis beforehand. Little to no error is tolerable because no simulation is possible. To offset any problems, extra caution, planning, and careful calculations were necessary.

**Calculations**

All calculations and procedures necessary for the design of the regulator came from the MAX1524 datasheet provided by Maxim Integrated Products. Mathcad did all the calculations performed to ensure correct values, as shown in the appendix. The set of calculations used come from the discontinuous conduction mode (DCM) portion of the datasheet in addition to the section for setting the output voltage and calculating the maximum duty cycle.

$V_{inmin} := 2.5V$ $t_{onmin} := 0.5 \cdot 10^{-6} s$ $V_{out} := 5V$ $I_{loadmax} := 500mA$

$V_D := 0.5V$ $I_{peak} := 1A$ $I_{out} := 500mA$ $t_{ss} := 3.2ms$ $V_{FB} := 1.25V$ $R_2 := 65k\Omega$

$V_{inmax} := 3.3V$ $t_{onmax} := 0.5 \times 10^{-6} s$ $V_{in} := 3V$ $R_L := .005\Omega$ $L_1 := 380 \cdot 10^{-9} H$

$C_{out} := 12 \cdot 10^{-6} F$ $Q_g := 11 \times 10^{-9} C$ $I_{load} := 300mA$

**Setting the Output Voltage**

$$R_1 := R_2 \cdot \left( \frac{V_{out}}{V_{FB}} - 1 \right)$$

$R_1 = 1.95 \times 10^5\ \Omega$

**Finding the Maximum Duty Cycle**

$$DutyCycle_{MAX} := \frac{V_{out} + V_D - V_{inmin}}{V_{out} + V_D} \cdot 100$$

$DutyCycle_{MAX} = 54.545$

$$\frac{R_1}{4} = 4.875 \times 10^4\ \Omega$$

**Calculating the Peak Inductor Current**

$$I_{PEAK} := \frac{1.15(V_{out} + V_D)}{V_{in}} I_{loadmax}$$

$I_{PEAK} = 1.054\ A$

**Selecting an Inductor**

$$L_{IDEAL} := \frac{V_{inmin} \cdot t_{onmin}}{3 \cdot I_{PEAK}}$$

$L_{IDEAL} = 3.953 \times 10^{-7}\ H$

$L_{IDEAL} - .5 \cdot L_{IDEAL} = 1.976 \times 10^{-7}\ H$

$$P_{LR} := \left[ I_{load} \cdot \left( \frac{V_{out} + V_D}{V_{in}} \right) \right]^2 \cdot R_L$$

$P_{LR} = 1.512 \times 10^{-3}\ W$

$.33 \cdot 10^{-6} \cdot .2 = 6.6 \times 10^{-8}$

$.33 \cdot 10^{-6} + 6.6 \times 10^{-8} = 3.96 \times 10^{-7}$

$$\frac{R_1}{32R_2} \cdot V_{FB} + V_{FB} = 1.367\ V$$

**Switching Frequency**

$$f_{SWITCHING} := \frac{V_{out} + V_D - V_{in}}{t_{onmin} \cdot (V_D + V_{out})}$$

$f_{SWITCHING} = 9.091 \times 10^5\ \frac{1}{s}$

$$f_{LIGHTLOAD} := \frac{V_{out} + V_D - V_{in}}{t_{onmin} \cdot (V_D + V_{out})} \cdot \frac{I_{load}}{0.18 \cdot I_{loadmax}}$$

$f_{LIGHTLOAD} = 3.03 \times 10^6\ \frac{1}{s}$

### Output Capacitor Selection

$$C_{outmin} := \frac{I_{load} \cdot t_{onmin}}{0.005 V_{out}}$$

$C_{outmin} = 6 \times 10^{-6}\ F$

$$C_{outmax} := \frac{I_{loadmax} \cdot t_{ss}}{V_{out}}$$

$C_{outmax} = 3.2 \times 10^{-4}\ F$

### Resistance on the Capacitor

$$ESR_{coutmin1} := \frac{L}{C_{outmin}} \cdot \frac{I_{loadmax}}{V_{in}}$$

$ESR_{coutmin1} = 27.778 \frac{m^3}{s}$

$$ESR_{coutmin} := 60 \cdot 10^{-3} \cdot \frac{V_{FB}}{I_{PEAK}}$$

$ESR_{coutmin} = 0.071\ \Omega$

$V_{RIPPLE} = 0.3 I_{PEAK} ESR_{coutmin}$

$V_{RIPPLE} = 0.023\ V$

### Feedforward Capacitor

$$C_{FF} := 3 \cdot 10^{-6} \cdot \left( \frac{1}{R_1} + \frac{1}{R_2} \right)$$

$C_{FF} = 6.154 \times 10^{-11} \frac{1}{\Omega}$

$$C_{FB} := \frac{1}{40} \cdot C_{FF}$$

### MOSFET Selection

$I_{GATE} := Q_g \cdot f_{SWITCHIN}$

$I_{GATE} = 0.01\ A$

### Feedback Capacitor

$$Ripple_{FB} := V_{RIPPLE} \frac{C_{FF}}{C_{FB} + C_{FF}}$$

$Ripple_{FB} = 0.022\ V$

Note: $Ripple_{FB}$ should be 25mV to 40mV. Cannot get calculation to work out due to the limits of the function.

### Current on the Diode

$I_{DIODE} := \sqrt{I_{out} \cdot I_{peak}}$

$I_{DIODE} = 0.707\ A$

**Circuit Diagrams**

Below (**Error! Reference source not found.**) is a diagram of the MAX1524 circuit that has become the voltage regulator for the Biobattery. Although the values for L1, R1, R2, $C_{FF}$, and C2 are different to give the regulator the ability to take a lower input voltage, the rest of the topology remains the same. One further note is that the pin marked SHDN needs a connection to $V_{CC}$.

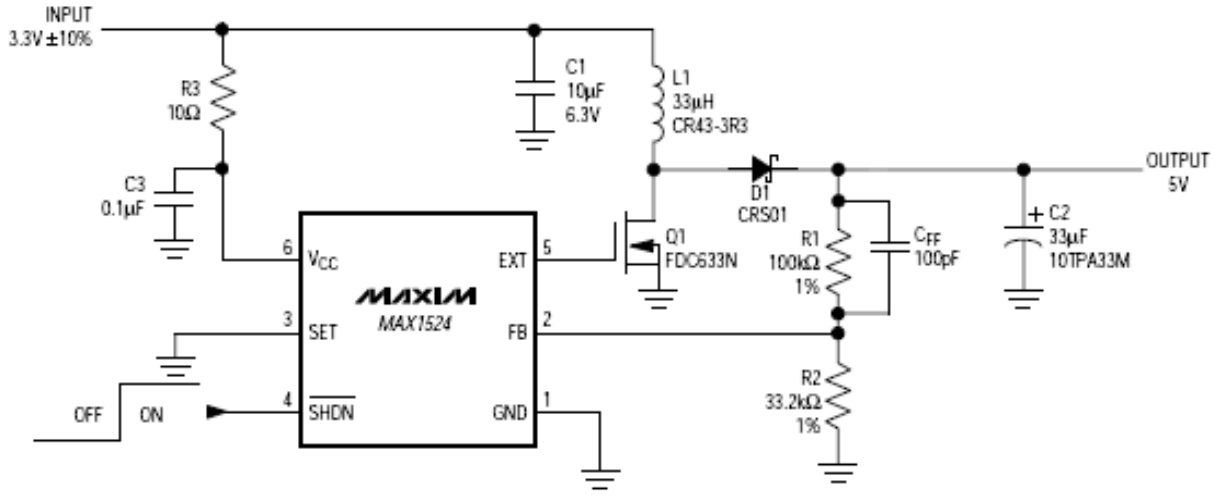



Figure 22 - Regulator Circuit Diagram

### 6.4.9. USB Power Switch

The MAX1607 IC generates the over-current signal. This IC has multiple purposes, which include USB compliance, provision of other features such as over-current, and short-circuit protection to a loaded system. The regulated output connects to the input of this IC before it is possible to attach a load. To provide power to a USB device, some specifications need fulfillment, and this IC will meet those requirements. It will not change the input voltage because its main objective is to protect the system from over-current withdraw by a load.

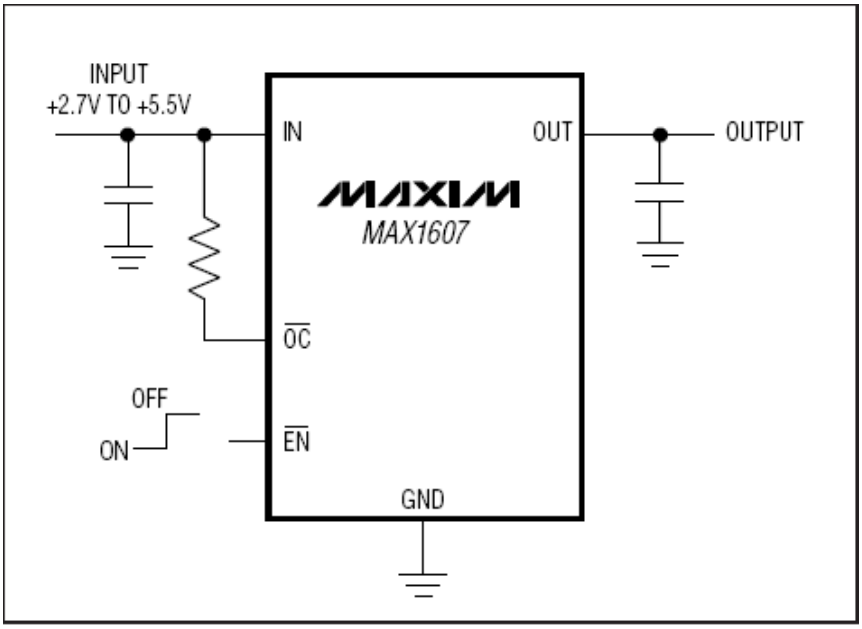



Figure 23 - MAX1607 Circuit Layout

### 6.4.10. Monitoring System

The main objective of the monitor is to oversee the system status and relay the relevant information about the bio-battery system to a user. Four main variables need monitoring: the voltage output from the MFC, the feed/waste level, the temperature of the surrounding, and current drawn by the load. These four parameters will help to determine the stability of the Biobattery system. If any of these parameters fail or go outside of the specified range, the user will need to know that a malfunction occurred.

**Design consideration/criteria**

The most important criterion in designing the Monitor is that the power consumption by this unit has to be at minimum. Monitor will be sharing the power generated by MFC with Regulator. The bulk of the power generated needs to go towards the load, which will connect to the bio-battery to ensure that the Monitor itself needs little power to operate. Since the Biobattery is a green product, it is essential that any component used for monitoring the

system be RoHS compliant and lead free. The information relayed to the user must display in a user-friendly manner, so he/she can understand the system status.

**Design alternatives and analysis**

There are different ways to monitor each of the variables mentioned above. The next section will discuss each of those variables in detail

### Voltage output from MFC

An analog or a digital voltmeter can monitor the voltage. An analog voltmeter will use no power from the MFC. It is a mechanical system with coil and spring, which will produce a deflection and corresponding reading. One can calibrate the device to read the voltage output level from the MFC. A digital voltmeter is more complex. It will require at least three main components: an ADC, a micro-controller and an output unit such as an LCD, LED, etc. Since these are all electrical components, they will consume power. However, power consumption by the digital circuit is very little. A typical IC consumes power in the milli-watt range. A digital unit is more reliable than an analog unit is. Both the analog and digital unit need shielding in some way from the water contained in the Biobattery system. An analog unit will rust over time from moisture just as the digital circuit will fail, too.

### Feed/waste level

The feed and waste need monitoring to let the user know when it is time to change the feedstock as well as remove the waste from the system. Sensors such as level or pressure sensors are attachable to the feed and waste, which will tell the user to take some action when the feedstock is low or the waste needs draining. The signals generated by the sensors will need processing—hence, also requiring power from some source, which is the MFC in this case. Another way to monitor the feed and waste levels is simply to use clear transparent feed and waste level. A user can easily replace the feed when the feedstock stored in a clear bag is about to go empty. Similarly, filling a small-sized glass ring when the waste level rises to a certain level will tell the user the waste is about full and needs draining.

### Temperature

The temperature of its environment heavily influences the feeding behavior of Geobacter. The optimum temperature range is 20-35 Celsius for activity of these bacteria. The temperature beyond these ranges will not produce the required voltages. The user needs to know when the temperature is not within the range. Implementing an analog thermometer gauged to a predetermined reading can work for this purpose. The alternative is to use a

temperature sensor, which will require some signal processing and will consume power in order to do so.

## Current

The Biobattery can only produce so much current to provide for a load. When the load starts to draw current above the level that the Geobacter can keep up with, the performance of the bio-battery will degrade significantly. Therefore, the system needs protection from an over-current withdrawal by a load. Using a fuse can help regulate the current being drawn from the system. Whenever an overload happens, the fuse will break, stopping the flow of current. The user will then need to replace the fuse. Another way to protect from the over-current condition is to implement a circuit that will detect the current limit reached and shut off the power provided to the load. Once the user removes the load, the Biobattery will function normally without having to change any component.

**Decision**

An analog system is comparatively easier to implement in order to monitor the system status. It will also not consume any power from the MFC, which is the biggest advantage. The disadvantages in using a analog method is that it is not very user friendly, especially when the user will have to use some form of judgment to monitor the system. The user will not need to read various measurements and act on it. Each of the measuring units is individual, which will increase the cost of the system. It is aesthetically unpleasing as well. One major problem is the use of thermometer, since it will contain mercury. The product designed has to be environmentally friendly, and the use of mercury goes against this. In addition, the replacement of a fuse every time the system overloads does not to the user, especially when he/she will need to buy it.

A digital system can lose many of the disadvantages of the analog meters. First, to reduce the size and cost, all of these different units are combinable into one unit. The digital system is more precise and is more user-friendly. The main concern with the electrical system is the power consumption; however, the digital components today are very efficient and use very little power. The advantages of using a digital system hugely outweigh the one disadvantage, which is negligible. The performance of a digital system is much better than an analog system performing the same function. Even though it is more effective to use digital system to monitor voltage, temperature and current, it is still better to leave the monitoring of feed and waste to user. The change of feed and waste will occur over the period of days so that it does not need continuous monitoring. The use of sensors to measure feed and waste level will only complicate the system without providing any major benefit.

An AVR butterfly development kit will serve the purpose of monitoring module. The kit has built in temperature sensor, which simplifies the design.

**Implementation**

A digital system will carry out the monitoring of the system status and communicate the information to user.

### Hardware

The following figure shows a basic block diagram of a Monitor unit. The Monitor unit will take analog inputs from the MFC and temperature sensor to convert those inputs into digital signals using ADCs. A digital over-current signal will go to the microprocessor through one of its ports. After the microcontroller processes the signals, the LCD will display meaningful information.

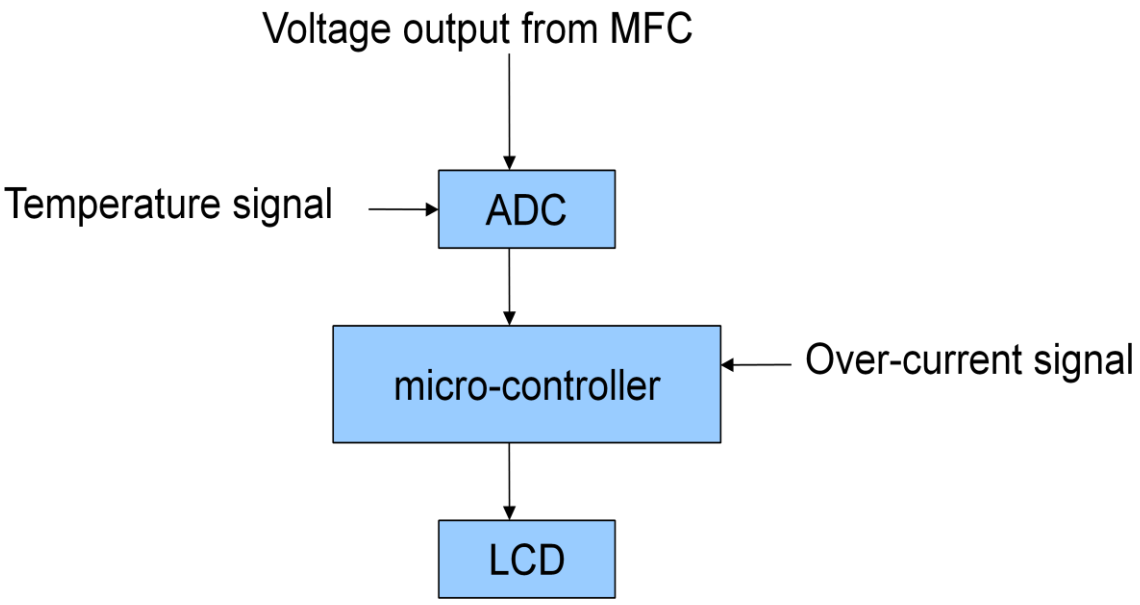



Figure 24 - Monitor Block Diagram

## AVR butterfly kit

The heart of the Monitor design is the AVR butterfly kit. The following block diagram shows the components included in the kit. Please see the appendix for the full specifications.

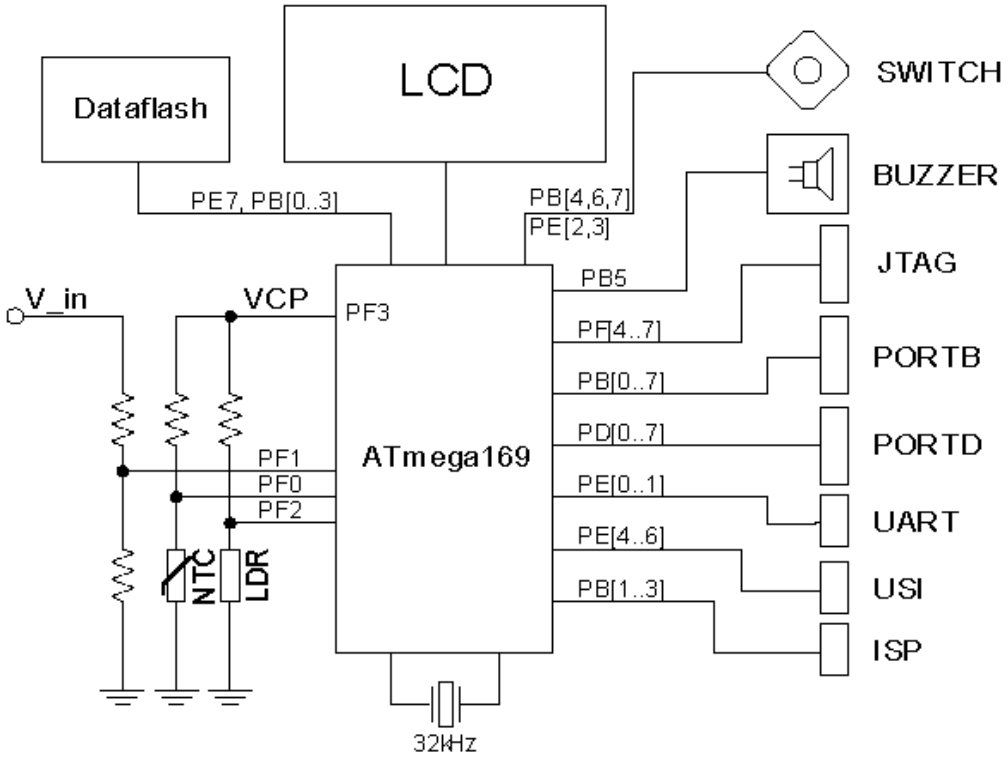



**Figure 25 - AVR Butterfly Kit**

The ATmega 169, the micro-controller, is the brain of the system. It is responsible for processing all the input signals and producing the appropriate output signals. The Appendix shows the pin layout of this micro-controller. Discussion of the ports and components used is here in detail.

## Input ports

### *PF1 port for voltage input*

The voltage output from the MFC will connect to the VIN port. This port connects directly to ADC1, which will convert the analog reading into a digital reading. This digital signal will then go to the micro-controller via the PF1 port.

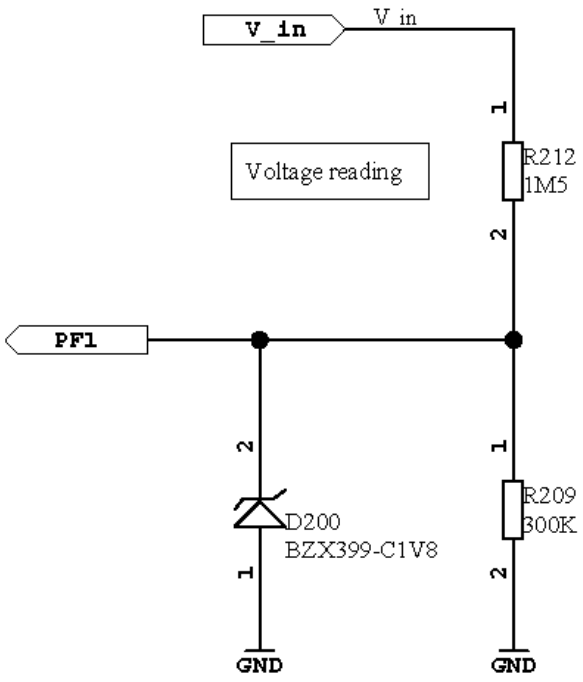



Figure 26 - Voltage Port Schematic

*PF0 port for temperature sensor*

The temperature sensor also connects directly to ADC0. As with the voltage reading, the temperature reading converts into a digital signal from an analog signal, which then forwards to the micro-controller via the PF0 port.

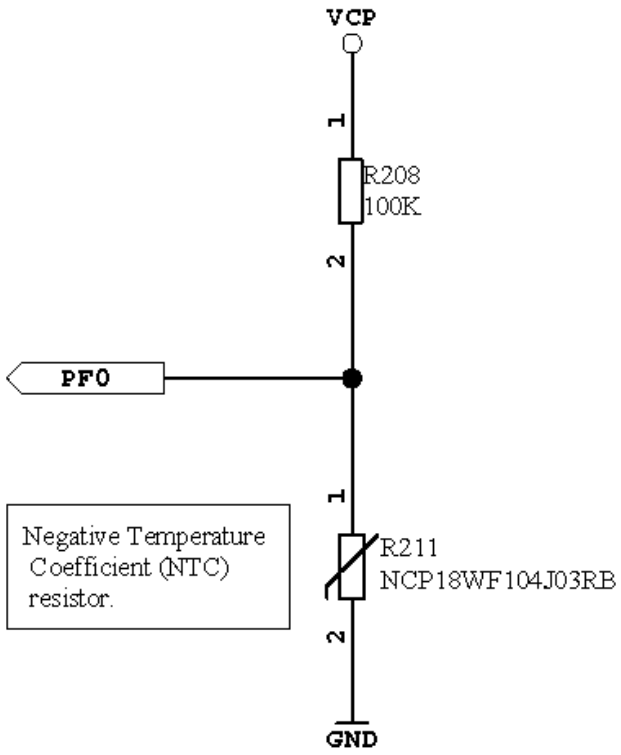



**Figure 27 - Temperature Sensor Schematic**

*USI port for over-current signal*

The over-current signal, which already is in digital form, goes to the USI pin PE4. This signal tells micro-controller that there is some error in the output of the system.

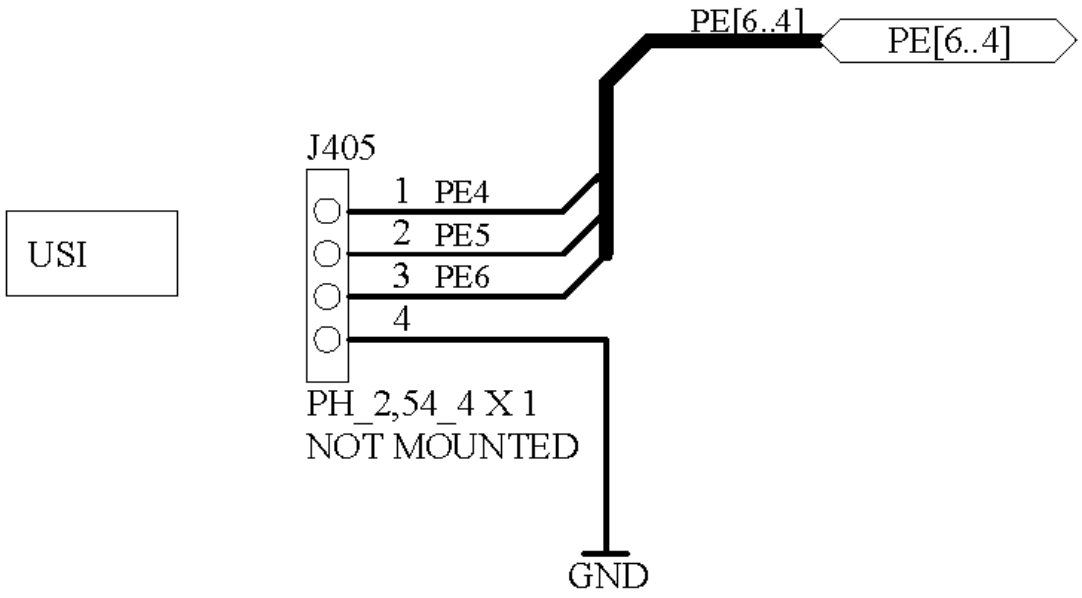



**Figure 28 - USI Port Schematic**

## Output ports

### *LCD*

The LCD connects directly to the micro-controller to display all the information in text.

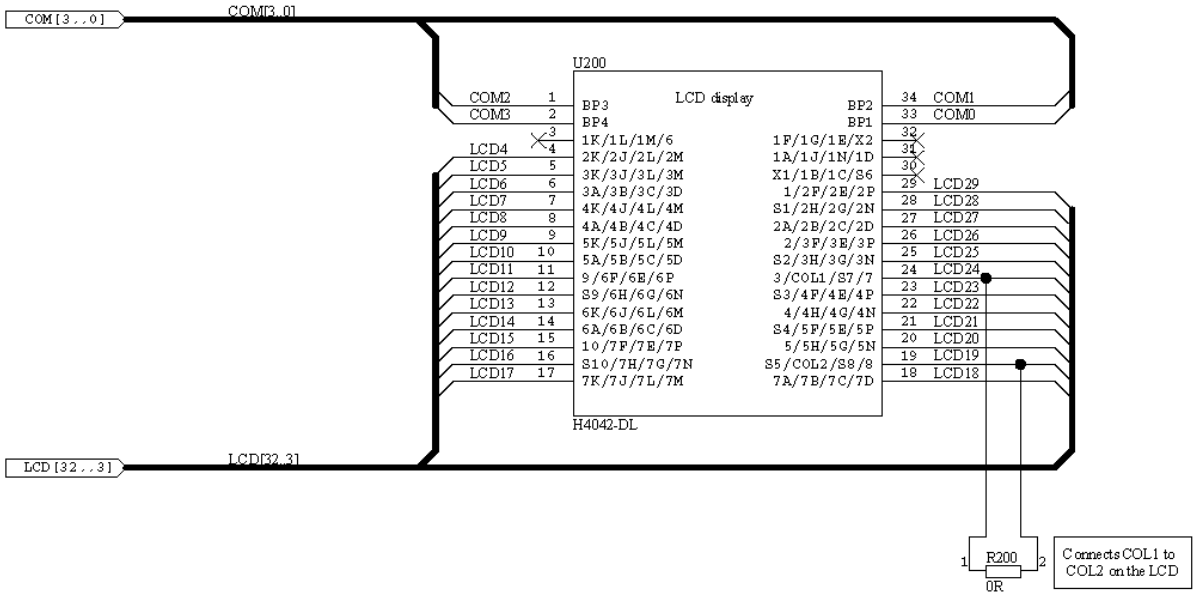



**Figure 29 - LCD Schematic**

*Port B for the enable signal to MAX1607*

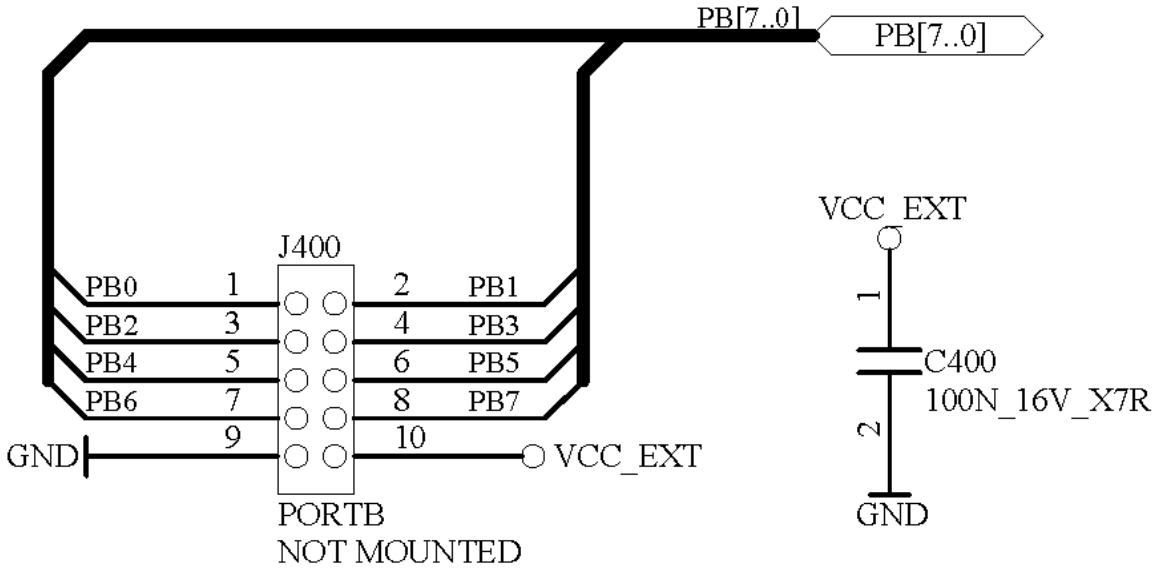



**Figure 30 - Port B Schematic**

Any time an over-current or short-circuit condition in the circuit occurs, it is necessary to disable the output so that no further damage happens to the system. Therefore, whenever the MAX1607 generates an over-current signal, the micro-controller sends back a signal to the Enable pin in the MAX1607 via PB0 of the AVR butterfly to disable the output. As soon as the system removes the over-current fault, the micro-controller sends another signal to re-enable the output of the system.

## Firmware

The micro-controller will perform the entire signal processing after receiving the inputs from its ports. Then, it will send the output to LCD. The software does this processing. The program, written in C, will control what the Monitor needs to do to meet the required condition. Producing a better response requires the proper handling of signals.

### *Voltage*

The voltage produced by the MFC connects to a 10 bit ADC. Using the following formula will calculate the voltage:

$$Vin = (ADC/1024)*Vref$$

After calculating the voltage, a display shows the level (Table 7 - Monitor Voltage Level Display) after a conversion takes place.

**Table 7 - Monitor Voltage Level Display**

| Vin | Display text |
|---|---|
| > 3.0V | Battery Good |
| 2.7V – 3.0V | Battery OK |
| 2.3V – 2.7V | Battery Low, No output! |
| <2.3V | Battery Flat, Feed now! |

The figure to the right () shows how the voltage coming out of the MFC compares to the appropriate output information to the user.

```
void Power_check(void){
    int voltage = 0;

    ADC_init(1);
    voltage = ADC_read();

    PGM_P statetext;

    if (voltage > PWR_MAX_V){ // battery is really really good
        statetext = PSTR("Batt Full");

    }else if (voltage> PWR_GOOD_V){ // battery is freshly charged
        statetext = PSTR("Batt Good");

    }else if (voltage>PWR_LOW_V){ // normal operation mode
        statetext = PSTR("Batt OK");

    }else if (voltage>PWR_MIN_V){ // only 12% left
        statetext = PSTR("Batt Low");
    }else {
        statetext = PSTR("Batt Flat");
    }
    LCD_UpdateRequired(TRUE,0);
    LCD_puts_f(statetext, 1);

    statetext = NULL;
}
```

**Figure 31 - MFC Power Check Function**

*Temperature*

The temperature sensor also connects to a 10 bit ADC. The temperature reading converts to equivalent voltage reading. Then the optimum range for temperature displays as shown below:

**Table 8 - Monitor Temperature Level Display**

| **Temperature** | **Display text** |
|---|---|
| 20° C – 35° C | Optimum temperature |
| Rest | Temperature out of range, battery performance low |

The following function (Figure 32 - Temperature Check Function) reads the temperature and determines whether the temperature is within the range.

```
char Temp_check(void){
    int temperature = 0;
    char temp_error = 'F';

    ADC_init(0);
    temperature = ADC_read();

    if (temperature > 574 || temperature < 396) {
        temp_error = 'T';
    } else {
        temp_error = 'F';
    }
    return temp_error;
}
```

**Figure 32 - Temperature Check Function**

### *Current*

The over-current signal directly connects to one of the input ports of micro-controller. Whenever this signal generates, it will trigger an interrupt and the micro-controller will send a signal to disable the output. Upon removal of the condition that caused the interrupt, a re-enable of the output will occur.

### *Main Program*

These three different signals have a specific priority. The following flowchart (Figure 33 - Monitor Flow Chart) shows the control logic of the program. Activating the monitor will cause it to first check for any circuit failure. Then, the monitor will sense the temperature so that the Biobattery can operate at optimum conditions. After that, a voltage check happens to make sure the circuit produces enough power for the load. The Monitor will go to power save mode when the system is idle for a minute. Pushing the button once can activate the monitor.

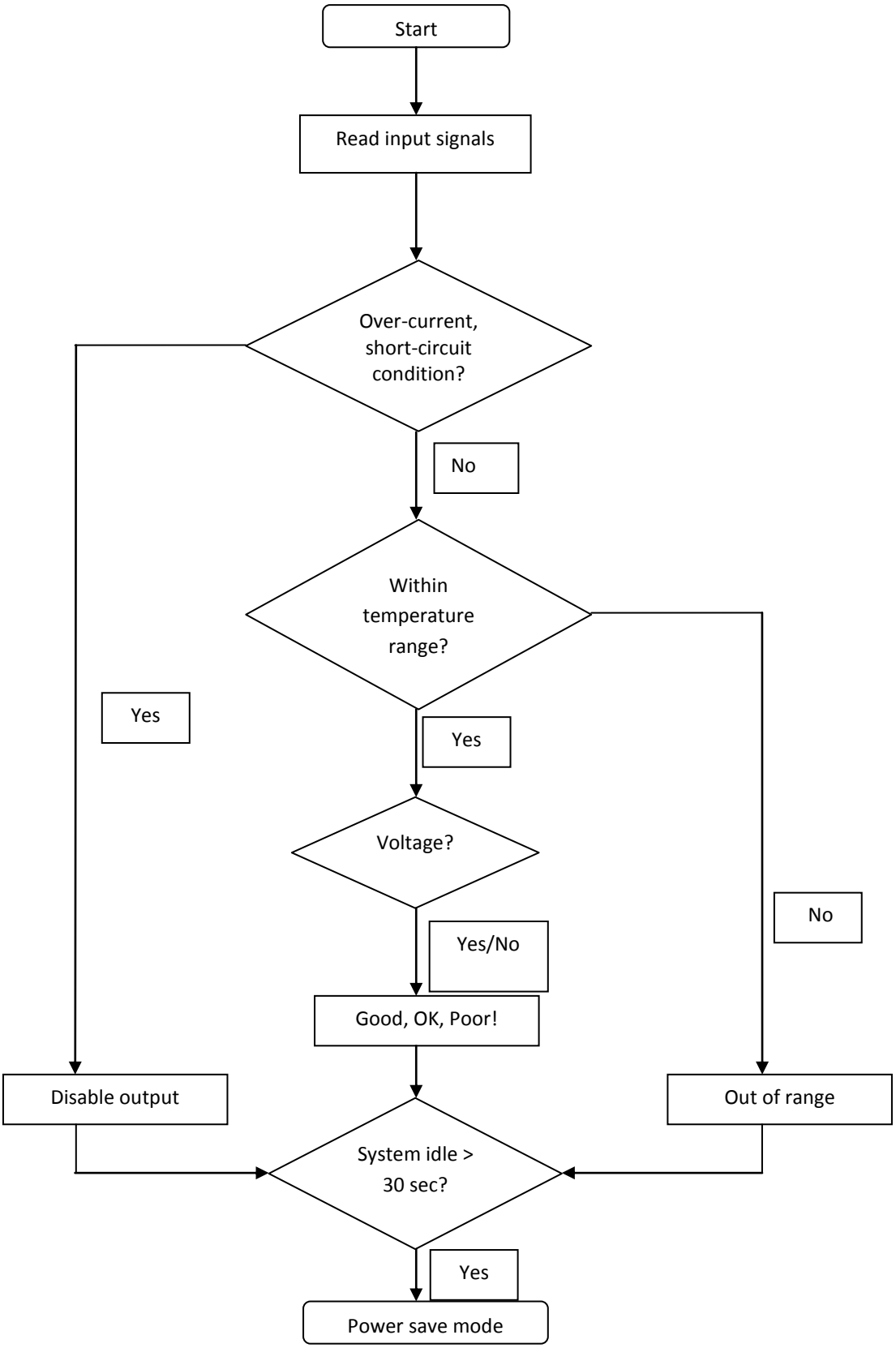



Figure 33 - Monitor Flow Chart

Here is the snippet of the main function that controls how the logic flows. As shown in the above flowchart, first, it checks for any circuit failure condition; then, it checks the optimum temperature range for MFC to operate; and finally it checks the voltage of MFC. Whenever the system sits idles for more than 30 seconds, it goes to power save.

```
int main(void)
{

    // Program initalization

    Initialization();
    sei(); // mt __enable_interrupt();

    char i;
    char input;

    for (;;){

        LCD_puts("ECOWATT",1);
        Delay(3000);

        if (!(PINE & 0x10))
        {
            LCD_UpdateRequired(TRUE,0);
            for (;;)
            {
                if (gPowerSaveTimer >= PowerSaveTimeout) break;
                LCD_puts("Circuit error",1);
                Delay(5000);
            }
        }

        LCD_Clear();
        LCD_UpdateRequired(TRUE,0);
        char temp_err = 'F';

        int temp_val ;
        ADC_init(0);
        temp_val = ADC_read();
        ADC2Temp(temp_val);
        Delay(3000);
        LCD_Clear();
        LCD_UpdateRequired(TRUE,0);
```

Figure 34 - Main function of Monitor Firmware Part 1 of 2

```
        for (;;)
        {
            temp_err = Temp_check();
            if (temp_err == 'T' ) {
                if (gPowerSaveTimer >= PowerSaveTimeout) break;
                LCD_puts("Temp out of range",1);
                Delay(5000);
            } else {
                break;
            }
        }

        LCD_puts("Optimum Temp",1);
        Delay(5000);

        LCD_Clear();
        LCD_UpdateRequired(TRUE,0);
        for(;;)
        {
            Power_check();
            Delay(4000);
            if (gPowerSaveTimer >= PowerSaveTimeout) break;
        }

        gPowerSaveTimer = 0;
        LCD_puts_f(PSTR("bye"),1);
        Delay(3000);
        SMCR = (2<<SM0) | (1<<SE);        // Enable Power-down mode
        asm volatile ("sleep"::);

        if(!(PINB & 0x40))                // press UP to wake from SLEEP
        {
            //PowerSave = FALSE;
            for(i = 0; i < 20; i++) // set all LCD segment register to the variable ucSegments
            {
                *(&pLCDREG_test + i) = 0x00;
            }

            sbi(LCDCRA, 7);               // enable LCD
            input = getkey();             // Read buttons
            gPowerSaveTimer = 0;       // reset seconds until sleep timer
        }
        SMCR = 0;                         // Just woke, disable sleep
    } //End Main loop
        return 0;
}
```

Figure 35 - Main function of Monitor Firmware Part 2 of 2

### 6.4.11. Fuel Cell Arrangement Alternatives and Analysis

Three options exist for the fuel cell arrangement. Like with a circuit, the arrangement has the potential to be in series or in parallel. Despite the continuing research, prior knowledge concerning the arrangement gives weight to each of the possibilities.

If the fuel cells are in series, the voltages of each cell will add together. However, a problem exists for serial connection. If one of the cell's output becomes unusable, it will short the circuit between the rest of the cells. In the final design, this has the potential to become difficult to repair. As a result, the entire battery can become crippled by one small malfunction, such as the death of the bacteria in one of the cells.

On the other hand, one can wire the fuel cells in parallel. While this option can possibly suffer from lower output, it has a higher guarantee for success. That is, it is less likely to short out from the failure of one fuel cell. The difference here is akin to how a string of lights works. If one goes out, they all go out versus having one go out while the others still work properly.

For the purpose of having a reliable current and voltage, the fuel cell arrangement decision needs careful consideration. Therefore, Ecowatt decided to use a combination of both parallel and series for the fuel cell arrangement. To expound, imagine a string of lights running in parallel with another string of lights. Each string of lights is in series by themselves; hence, if one light goes out, they all go out. Likewise, for the fuel cell arrangement, this result is undesirable. Now imagine that the lights are electrically connected at the ends so that the power source powers both light strands. Unfortunately, power between the strands becomes divided up so that the light given off from each strand of power is dimmer than if each strand were to receive power by itself from separate sources. Again, this result is undesirable for the fuel cell arrangement. Consequently, the design for the fuel cell arrangement requires a compromise between the two of these options, which leads to the third option, combining the two previous options. By running the strands of lights in parallel and connecting them between every light with a small resistance, each light will run in series with the next light while being in parallel with the next light on the other strand of lights. Also, by placing the resistance between the strands of lights, a scenario comes into play where the resistances and the lights will not cancel each other's effects out or be able to be reduced further. Similarly, by placing a small amount of resistance between the fuel cells and running the half of the fuel cells in series with the other half in parallel to that series, one can simultaneously receive the result of having a circuit in both series and parallel.

## 6.5. Test Results

### 6.5.1. Monitor

**Voltage**

A variable power supply connects to the VIN port of AVR Butterfly to test the voltage. In addition, an external multi-meter checked the actual output from the power supply. In varying the output from the power supply from 0V to 5V, the LCD displayed the power level as expected. The maximum voltage reading at Vin is 3V, and since our ADC performs 10-bit conversions, the accuracy of the voltage reading is $3V/(2_{10})$ or roughly 3mV.

**Temperature**

The temperature measured by the sensor is of the surrounding environment. The temperature reading occurred in different places, such as a basement, at room temperature, and outside on a sunny day. It was necessary to make a check between the temperature reading and a normal thermometer. In all three different environments at different times matched the readings of temperature by the Monitor.

**Over-current**

Testing the over-current condition was the biggest challenge. The best way to test for this condition was physically to simulate the over-current condition. The MAX1607 IC used to detect the over-current or short-circuits condition produces a logic high whenever this condition occurs. This logic high interrupts the micro-controller to take some preventive actions. In order to produce this interrupt, i.e. to send a logic high signal to PE4 pin, the desired pin had an external voltage applied to it, which interrupted the micro-controller as expected. With the triggering of the interrupt, the output of MAX1607 IC was disabled. The micro-controller sent a logic high signal to enable pin in MAX1607 IC via port PB0. Only in removing the interrupted event condition can the micro-controller send a logic low signal to enable the pin to reactivate the output.

**Power-save mode**

Every 30 seconds after the activation of the Monitor, the Monitor went into power save mode. This was easy to test by timing the event against a regular stopwatch, which showed that the Monitor went into power save mode each time. In order to see the difference in power consumption by the Monitor, it was necessary to the power consumption during active mode and the power saving mode. The power supplied to the Monitor was 3V. During active mode with text scrolling in the LCD along with the ADC measuring voltage and temperature, the power consumed was 2.13mW while during power-save mode; the power consumed by Monitor was 24µW. The ratio of power in active mode to power save mode is almost 90. This

clearly indicates that it is better to use power-save mode as much as possible. Therefore, activating the Monitor only when the user needs to know what is happening with the system rather than constantly monitoring the system is best.

### 6.5.2. Regulation

**Functionality**

To explain the operation of the regulator is important. An input into the voltage regulator circuit enters the circuit. Throughout the circuit are capacitors, each of which holds a charge before discharging over time. The only exceptions to this for functionality are the feed forward and feedback capacitors, which ensure proper switching and ripple at the feedback pin. If too much ripple occurs, the result will show at the output as well. Likewise, the inductor discharges over time as well, making the MOSFET act like a switch to prevent too much or too little charge from going to the output. The MAX1524 chip senses this so that the chip will boost the voltage when the voltage and current are enough to turn the chip on from its soft-start mode. If the chip senses too much voltage, a fault will occur, turning the chip and circuit off. The chip has an internal reference for voltage, maintained by the voltage switching at a frequency defined by the designer, which is achievable by setting the resistance in R1 and R2, according to the calculations provided by the datasheet from Maxim. Setting this voltage divider, the desired input and output voltages become programmable into the chip. To ensure that no large changes affect the performance of the circuit, one must put the MAX1524 in bootstrap configuration (a feedback loop to prevent problems in regulation) to prompt proper regulation. However, Ecowatt's circuit currently is not in bootstrap mode due to issues discussed later. Instead, the input goes directly to the $V_{CC}$ pin and the SHDN pin to bring the chip into operation quicker. Also, SET goes to ground for a lower on time at that pin, which achieves a lower range for the duty cycle to handle any given load. This is more suited to smaller increases in the voltage in terms of how much it needs an increase. In addition, by adding the diode on the output side of the circuit, current will not backtrack into the input of the circuit, which will stop any harmful activity to the bio-battery cells.

For the actual testing of the electrical portion, several characteristics were of interest to ensure the quality of the voltage regulator. These were the input versus output for both the voltage and the current, the output for any given input with a varied load, the power dissipation, the current and voltage at any given pin, and the current load efficiency. By testing specifically for these values, one can ascertain the functionality of the voltage regulator. In particular, the input versus output will show the voltage regulator's functionality and output. Next, by varying the load (i.e., adding a resistors of different values), one can test how the voltage regulator responds in terms of voltage and current. Another important aspect of testing is the amount of power dissipated by the circuit. While this measure relates to the circuit’s

current load efficiency, power dissipation gives a better idea of how much power the circuit uses in terms of actual numbers, not a percentage like the current load efficiency. In addition, the current load efficiency is dependent on a varied current load while the power dissipation seeks the maximum amount of power consumed by the circuit. Testing the current and voltage occurred at all pins in order to understand how the circuit works and if it works for that matter.

Given time constraints, not all of these values became available for testing. However, with the exception of varying the load for any given input, all values became available during testing. Testing was accomplished using a LeCroy WaveJet 322 Oscilloscope, two LeCroy PP010 cables, one Graymark Model 803 Solid State Power Supply, two Heath Built Model EUW-30 resistor boxes, Pomona Electronics connectors, and M Deller alligator clips. Attaching the circuit to the power supply and oscilloscope, one varied the voltage by increments of 0.25V to 3.0V. Using the resistor boxes, the power that the resistors were capable of handling increased from a quarter of a Watt to one Watt. Yet, to increase the voltage much beyond 3V will cause the resistors to burn out. To prevent this from happening, the testing only occurred from lower voltages up to 3V. Testing each part and pin on the board, one probed all values for the circuit and calculated those values (i.e., power and efficiency) that needed calculation, as shown in the appendix. The efficiency of the circuit followed a bell-shaped curve. This was despite the fact that the voltage increased with every check of the electrical characteristics on the circuit. In probing the circuit, one must note that the noise generated by the circuit on both the input and output sides was quite significant. While this is undesirable, accurate measurements were necessary. In measuring the circuit with the oscilloscope, somewhat of an oddity came to attention: the amperage-increased amp for voltage as the voltage increased. Hence, further study is necessary on this matter to confirm the validity of the measurements. For this reason, further refinement of the circuit is necessary, which is in a later section. Nonetheless, some charts of the results of the testing are below.

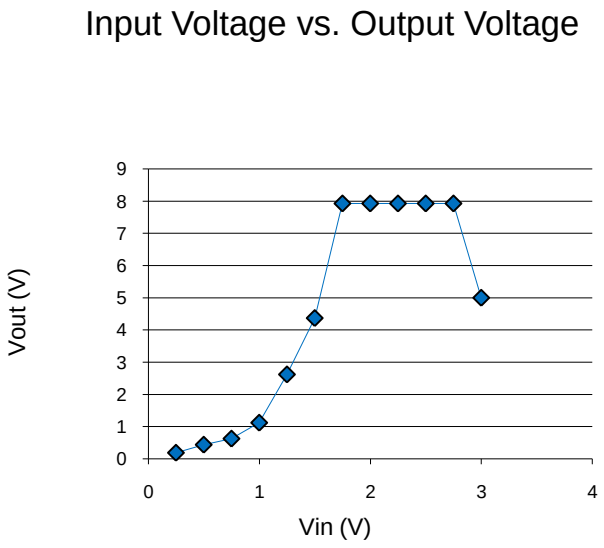



Figure 36 - Input Voltage vs Output Voltage

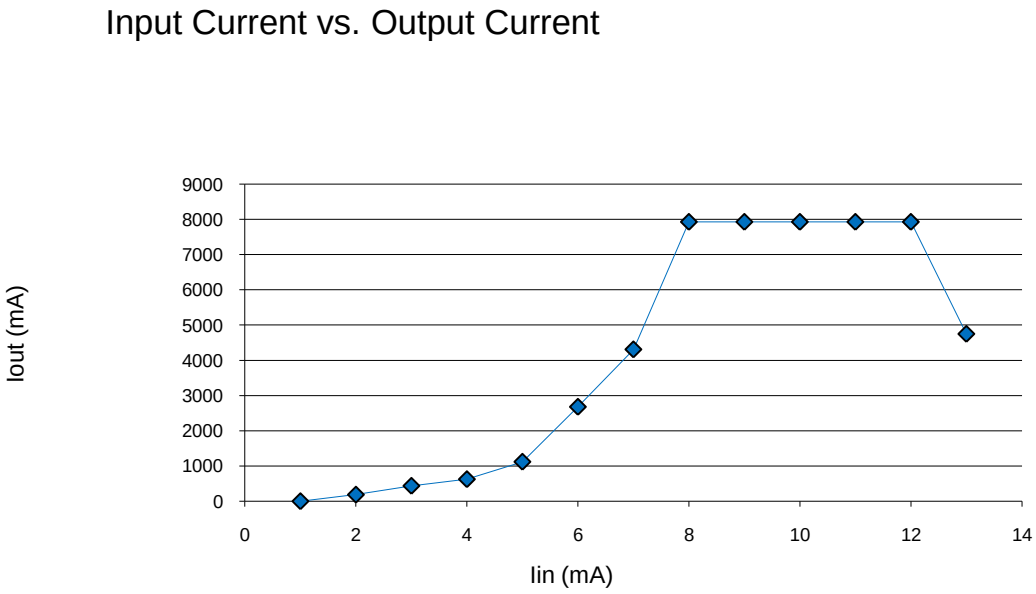



Figure 37 - Input Current vs Output Current

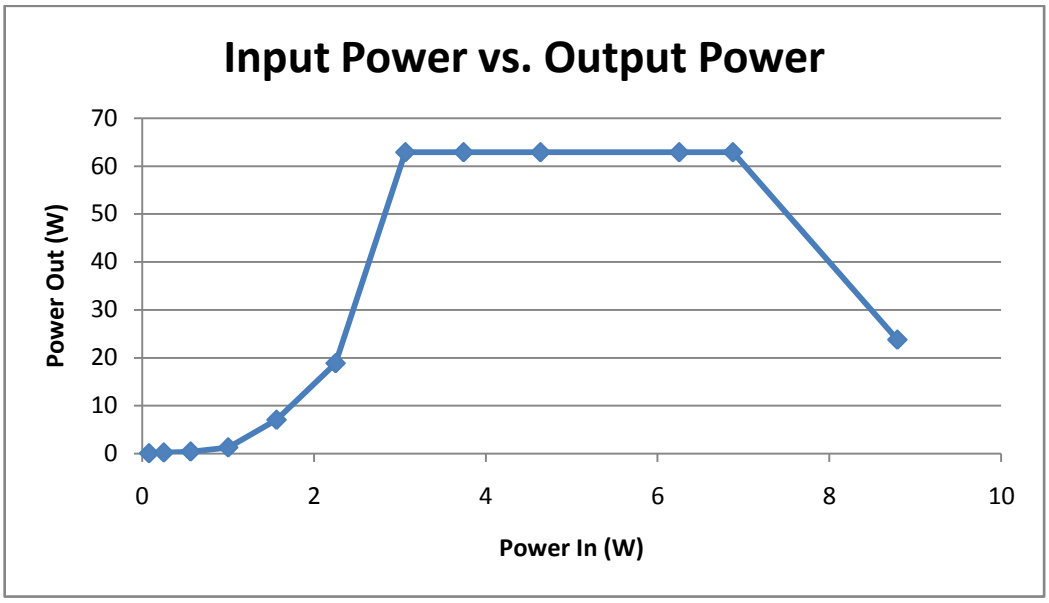



Figure 38 - Input Power vs Output Power

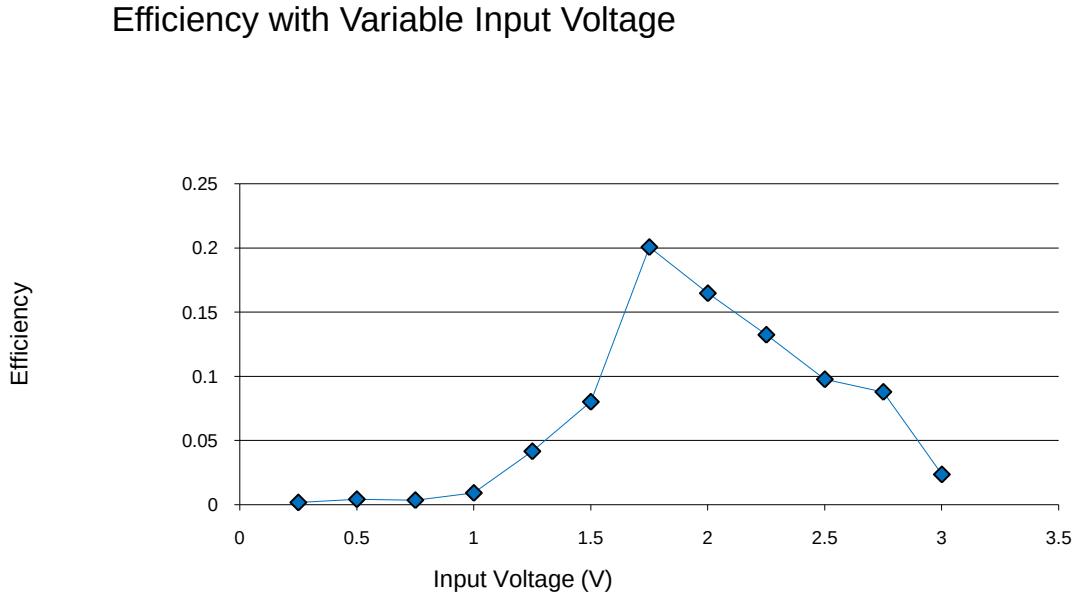



Figure 39 - Efficiency with Variable Input Voltage

Notably, the power curve for efficiency follows an almost bell-shaped curve, meaning that the circuit has an optimum efficiency around 1.75V. Since the placement of the peak in efficiency is lower than expected, doing the changes recommended in the next section will change this measurement.

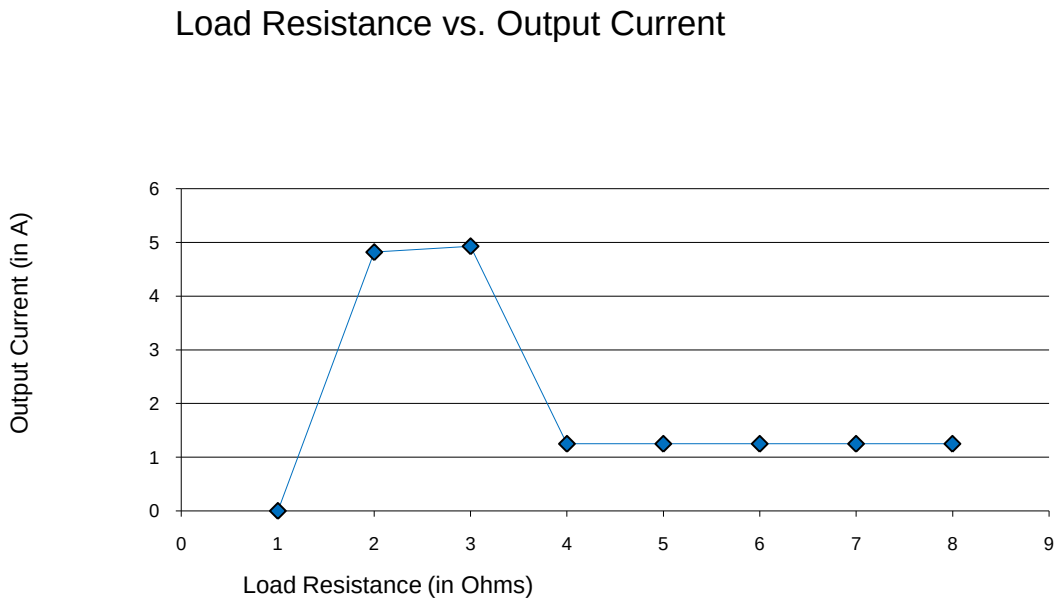



Figure 40 - Load Resistance vs Output Current

Seeing how the output current reacts to the load resistance, further investigation is necessary as to why this occurs.

**Performance**

Currently, the voltage regulator refuses to give the desired output of 5V. The regulator regulates to a voltage of 7.93V instead. Research is ongoing as to the reason why. Several reasons for why the regulator will not behave as desired have come to attention. One of these reasons includes the fact that a miscalculation occurred, resulting in the inductor being too small by one degree of magnitude. Assistance from Maxim's support team helped to point out this error. In addition, the resulting current from going through the inductor is too small, given that the power supply used only has a current of approximately 300mA. Another problem includes the fact that the output from the regulator is not a flat response. Instead, the output is noisy and sometimes oscillates like a sine wave. This problem will go away with the introduction of a feedback capacitor, attached to the connection of the voltage divider of R1 and R2 with the FB pin on the MAX1524. By placing the capacitor here, one can decrease the ripple on the output. Furthermore, the MAX1524 chip is in a non-bootstrapped configuration right now. By putting the MAX1524 chip in a non-bootstrapped configuration, the chip gave a higher output, which was contrary to all expectations, especially given what the datasheet for the MAX1524 stated. By fixing the inductor, increasing the current, and adding a feedback capacitor, putting the chip configuration back into bootstrap mode will become ideal. The idea is that by doing so the chip will operate according to plan. If this plan does not work, further

research will become necessary to offset any potential problems. Consulting with Professor Brouwer has shown that by rewiring the SHDN pin to the input instead of $V_{CC}$, the voltage regulator can perhaps achieve the desired voltage. While the regulator responds to varying resistor loads on the output side, the amount of response is minimal with respect to the amount of resistance put on the output.

# 7. Conclusion

## 7.1. Lessons Learned

The most valuable lessons we learned from this project came from the aspects we found most difficult. The interdisciplinary nature of this project and team members challenged communication styles and stretched individual vocabularies. Group communication was essential for each section of the design in the decision making process and prototype debugging. Multiple different cultural expectations were involved in the team members that did not merge flawlessly. Active listening is a tool that was developed though this project. Electrical and mechanical prototyping should be started in the first semester. Prototyping should be initiated as soon as possible concurrently with continuing research.

## 7.2. Future Work

Suggested future MFC work includes additional testing of different thicknesses of natural cellophane membrane to optimize proton flux. Only one thickness of cellophane was tested. The thicker membranes could have a decrease in proton flux but increased mechanical strength. Anode cube fabrication methods could be improved by reducing the unused side surface area. The anode cube currently has sides of 5 in with a 2in membrane. If the overall dimensions were decreased with the membrane area kept constant, the power to volume ratio would be significantly increase. The possibility of Platonized cathode side electrodes would eliminate the need for a cathode tank of water and significantly decrease the overall weight of the system. Platonized electrodes are approximately 15 times the cost of carbon cloth, our chosen electrode.

For the regulation portion of the Biobattery, given that many revisions occurred throughout the designing process, what came across, as a simple task at first became quite an intricate and timely process. Understanding and achieving the goals of Team Ecowatt in a timely manner became important. Therefore, conciseness and streamlining for the regulation part was essential to its success. In essence, this meant that the regulator needed thorough and quick design and testing to meet the deadline and fulfill the needs of the design criteria. This being true, given more time, better testing will become a possibility. Furthermore, the replacement and repair of parts and performance on the circuit will go into effect. Doing so will

fix the behavior of the circuit. Hence, this repair of trying to make the output out of the circuit behave correctly is necessary to any further considerations for improving the circuit design.

## 8. Acknowledgements

We would like to acknowledge the following people for their support throughout the duration of this project. Without their assistance and support we would not be able to undertake such a multi-disciplinary process.

- **Professor Ray Hozalski**, Civil Engineering, University of Minnesota – Twin Cities, for samples/supplies of electrodes, membranes, and information on MEAs.
- **Chris Harrington**, Graduate Student Researcher, University of Minnesota – Twin Cities, for help with implementation procedures.
- **Professor John Wertz,** Biology Department, for assistance in Microbiology growth and experimentation.
- **Professor J. Aubrey Sykes**, Engineering Department, for his ongoing role as the senior design advisor and for all of this feedback about our project.
- **Professor Randall Brouwer**, Engineering Department, for supplying VHDL code for ADC interface.
- **Sam Brower**, Media Productions Calvin Alum, for various visual design and photographic assistance.
- **Bob DeKraker**, Engineering Department, for logistical support with procurement of circuit components.
- **Rich Huisman**, Chemistry Department, for assistance with salt bridge supplies.
- **Lori Keen**, Biology Department, for assistance in biological procurement and lab support.
- **Professor Walter Rawle**, Engineering Department and Senior Design Team Mentor, for meeting with our team and assisting us with the in progress reviews.
- **Professor Gemma Reguera**, Michigan State University, for providing technical information and expertise.

# 9. References

"1210 MLCC Caps Datasheet." Digi-Key. Mar. 1998. Maxim Integrated Products. 23 Apr. 2008 <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=399-3297-1-ND>.

"4922(R) Series Datasheet." Digi-Key. Apr. 2004. API Delevan Inc. 23 Apr. 2008 <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=4922R-391L-ND>.

**Brown, Marty.** "Switchmode Power Supplies." Free Radio Berkeley. Sep. 1999. 6 Dec. 2007 <http://www.freeradio.org/documents/Switching_Power_Supply.pdf>.

"Chip Monolithic Ceramic Capacitors Catalog Datasheet." Digi-Key. 14 Dec. 2005. Murata Electronics North America. 23 Apr. 2008 <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=490-1346-1-ND>.

**Fabaey, K.; Boon, N. Siciliano, S. D.; Verhaege, M.; Verstraete, W.** Biofuel cells select for microbial consortia that self-mediate electron transfer. *Appl. Environ. Micronil.* **2004,** 70, 5373-5382.

"FDC633N Datasheet." Digi-Key. Mar. 1998. Fairchild Semiconductor. 23 Apr. 2008 <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=FDC633NCT-ND>.

**Goldwasser, Samuel M.** "Notes on the Troubleshooting and Repair of." Sci.electronics.repair.FAQ. 11 Feb. 2007. 6 Dec. 2007 <http://www.repairfaq.org/sam/smpsfaq.htm>.

Groff, Abbie. "Building a Two-Chamber Microbial Fuel Cell." Microbial Fuel Cells: From Waste to Power in One Step! The Logan Group. 4 Dec. 2007 <http://www.microbialfuelcell.org/>.

**Logan, Bruce E., Bert Hamelers, Rene Rozendal, and Uwe Schroder**. Microbial fuel cell: Methodology and technology. n.d.

**Lovely, D. R.,** Microbial Energizers: Fuel cells that keep on going. Microbe (Volume 1, Number 7), 2006

**Lovely, D. R., J. F. Stolz, G.L. Nord, and E. J. P. Philips.** Anaerobic production of magnetite by a dissimilatory iron-reducing microorganism. Nature (London) **330**:252-254, 1987.

MAX1522, MAX1523, MAX1524 Simple SOT23 Boost Controllers. 2001. Maxim Integrated Products. 23 Apr. 2008 <http://www.maxim-ic.com/quick_view2.cfm/qv_pk/2451>.

"MA3ZD12 Datasheet." Digi-Key. Apr. 2004. Panasonic - SSG. 23 Apr. 2008 <http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=MA3ZD1200LCT-ND>.

**Rabaey, K.; Lissens, G.; Siciliano, S.D.; Verstraete, W.** A microbial fuel cell capable of converting glucose to electricity at high rate and efficiency. *Biotechnology Letters*. **25**: 1531-1535, 2003.

**Simpson, Chester**. "Linear and Switching Voltage Regulator Fundamentals." National Semiconductor. National Semiconductor. 6 Dec. 2007 <http://www.national.com/appinfo/power/files/f4.pdf>.

"TPS62100 datasheet." DatasheetCatalog.com. Dec. 2000. 6 Dec. 2007 <http://www.datasheetcatalog.com/datasheets_pdf/T/P/S/6/TPS62100.shtml>. Path: Go to the website link and click on "TPS62100 datasheet" in middle of the screen to the right.

Underwriters Laboratories Inc. UL. 5 Dec. 2007 <http://www.ul.com/>.

**Wuidart, L**. "TOPOLOGIES FOR SWITCHED MODE POWER SUPPLIES." STMicroelectronics. 1999. STMicroelectronics. 6 Dec. 2007 <http://www.st.com/stonline/books/pdf/docs/3721.pdf>.

# 10. Appendices

10.1. Appendix I - Biobattery Assembly Instructions ..... 84
10.2. Appendix II- Bladder Filling Procedure ..... 87
10.3. Appendix III - MFC Testing Data ..... 88
10.4. Appendix IV - Product Information Sheet of ATCC® 53774 ..... 93
10.5. Appendix V - Membrane Testing Data ..... 94
10.6. Appendix VI - Nafion 117 Information sheet ..... 95
10.7. Appendix VII - Software Tools ..... 96
10.8. Appendix VIII – ATMega169 Specifications ..... 98
10.9. Appendix IX – Monitor Source Code (Written in C) ..... 100
10.10. Appendix X – Regulator Test Results ..... 124

## 10.1. Appendix I - Biobattery Assembly Instructions

**50mL Test Tubes with Salt Bridge, No Autoclave – Not Final Design**

Note: Prepared Bacteria Test Tubes from this procedure expire in about a week, but steps 1-3 do not expire and can be completed in large quantities in advance.

**Testing instruments:**

- Multimeter
- Resistors
- Cables
- wires
- alligator clips
- Oscilloscope
- Data logger

**Procedure:**

1. Prepare the wire and electrode. *{5 min / tube + Epoxy Curing Time}*
    a. Obtain a 10” wire with good conductivity (non-copper).
    b. Using a box cutter, cut a portion of electrode that will fit into the test tube (about 1cm X 0.5cm X 3cm, larger electrodes should result in higher power output).
    c. Attach the electrode to the end of the wire using a conductive epoxy or cement.
2. Prepare the cap (One for each desired bacteria-filled test tube, up to 16). *{20 min + 5 mins / tube, Requires Rich Huisman}*
    a. Obtain a rubber stopper of size 4.
    b. Insert a custom bent glass tube.
        i. Ask Rich to bend a glass tube (~18”) into three sides of a square.
        ii. Ask Rich to help create a hole in the rubber stopper for the custom bent glass tube.
    c. Insert the wire into the cap.
        i. Pierce the cap with a 16 gauge needle.
        ii. Thread the wire through the needle, leaving enough wire for the electrode to hang suspended just above the bottom of the test tube.

iii. Remove the needle, leaving the wire in the cap.

3. Prepare the Salt Bridge (Repeat for each desired bacteria-filled test tube). *{20 mins + 5 mins / tube}*

    a. Boil 30mL de-ionized water in a 100mL beaker with a stir-bar.

    b. Stir the solution and add ingredients.

        i. Add 1.5g KCl.

        ii. Add 2.5g Ultra-Pure Agar.

    c. Using a 16 gauge needle and a 10mL syringe, quickly add the Agar solution to the custom bent glass tube (from part 2) with as few bubbles as possible. The Agar solution will cool quickly and solidify.

4. Prepare Media Solution (No Autoclave). *{45 mins}*

    a. Boil 100mL de-ionized water in a 250mL Erlenmeyer Flask.

    b. Add 3.425g Sodium citrate (13.7g/L).

    c. Cool to room temp using ice bath.

    d. Adjust the pH to 6.0 using 10M NaOH.

    e. Add to the solution.

        i. Add 150mL de-ionized water.

        ii. Add 0.625g NaHCO3.

        iii. Add 0.0625g Ammonium Chloride.

        iv. Add 0.15g NaH2PO4H2O.

        v. Add 0.025g KCl.

        vi. Add 1.7g Sodium Acetate.

    f. Degas the media in a 1L Erlenmeyer Flask with Side Arm

5. Prepare the test tubes (Repeat for each desired bacteria-filled test tube). *{5 mins / tube}*

a. Using the nitrogen tank, blow the air out of a test tube.

b. Using an automatic pipette, transfer 15mL of prepared Media Solution to the test tube by quickly removing the cap and transferring the Media Solution.

c. Continue to blow the air out of the tube with the nitrogen tank and immediately close the cap again.

d. Add 7mL carbon dioxide.

e. Mix the gasses by shaking the test tube.

f. De-pressurize the test tube by quickly penetrating the cap with a needle. Remove the needle as soon as the tube audibly de-pressurizes.

6. Add the Vitamin and Mineral Solution (Repeat for each desired bacteria-filled test tube). *{5 mins}*

    a. Using a syringe and needle, add 0.16mL Wolfe's Vitamin Solution.

    b. Using a syringe and needle, add 0.16mL Wolfe's Mineral Solution.

7. Add the bacteria (Geobacter Metallireducens) (Repeat for each desired bacteria-filled test tube). *{10 mins / tube}*

    a. Select a healthy culture (light greenish brown).

    b. Sterilize the top of the culture.

        i. Using a cotton swab, apply a layer of 95% ethanol to the top of the cap on the culture solution test tube.
        ii. Using a Bunsen burner, ignite the entire top of the cap, burning off all of the ethanol.

    c. Using a sterile needle, pierce the cap of the culture solution test tube.

    d. Withdraw about 2mL culture solution.

    e. Add the culture solution to the media solution test tube by piercing the cap and injecting the culture.

8. Place each Prepared Bacteria/Media Solution in a 30°C Agitator Bath. *{1 min}*

## 10.2. Appendix II- Bladder Filling Procedure

**IV Bag Preparation with Autoclave**

*Note: Prepared Media Solution expires in about a week unless autoclaved.*

1. Prepare Media Solution (1L media produced) {45 mins}
    a. Boil 400mL de-ionized water in a 250mL Erlenmeyer Flask.
    b. Add 13.7g Sodium citrate (13.7g/L).
    c. Cool to room temperature using an ice bath.
    d. Adjust the pH to 6.0 using 1M NaOH.
    e. Add to the solution
        i. 600mL de-ionized water
        ii. 2.5g $NaHCO_3$
        iii. 0.25g Ammonium Chloride
        iv. 0.60g $NaH_2PO_4H_2O$
        v. 0.10g KCl
        vi. 6.8g Sodium Acetate
    f. Degas the media in a 2L Erlenmeyer Flask with side arm.
    g. Bubble $N_2$ in the solution for 10 mins.
    h. Bubble $CO_2$ in the solution for 5 mins and quickly cork.
        i. Remove a small sample and test pH (5.5-7.5 acceptable)

2. Fill IV bags with Media (2 bags of 500mL) {25 mins + autoclave}
    a. Empty and Wash IV bags.
    b. Fill each IV bag with 500mL and close.
    c. Autoclave {overnight}
        i. Label and place in Biology Lab Core
        ii. Ask Lori Keen to Autoclave overnight

3. Add Vitamin and Mineral Solutions (Repeat for each IV bag). {5 mins}
    a. Using a syringe and needle, add 0.64mL of Wolfe's Vitamin Solution.
    b. Using a syringe and needle, add 0.64mL of Wolfe's Mineral Solution.

## 10.3. Appendix III - MFC Testing Data

| | mL Media | Concentration Sodium Acetate |
|---|---|---|
| First Setup | 10mL | .082 mol/L |
| Second Setup | 10mL | .082 mol/L |

First setup voltage measurements

| Date | # Day | OCV(v) | I(A) | Rint(ohm) |
|---|---|---|---|---|
| 11/19/2007 | 0 | 0.63 | | |
| 11/20/2007 | 1 | 0.3 | 0.00002 | 14900 |
| 11/24/2007 | 5 | 0.3 | 0.00002 | 14900 |
| 11/26/2007 | 7 | 0.26 | 0.00001 | 25900 |
| average | | | | 18566.67 |

Second setup voltage measurements

| Date | #hrs | OCV(v) | I(A) | Rint(ohm) |
|---|---|---|---|---|
| 11/27/07 4:30 PM | 0 | 0.2 | 0.000034 | 4882.352941 |
| 11/28/07 9:00 AM | 16.5 | 0.314 | 0.000011 | 27545.45455 |
| 11/28/07 1:30 PM | 21 | 0.323 | 0.000021 | 14380.95238 |
| 11/29/07 8:00 AM | 39.5 | 0.308 | 0.000008 | 37500 |
| 11/29/07 4:30 PM | 48 | 0.335 | 0.0000081 | 40358.02469 |
| 11/30/07 2:00 PM | 69.5 | 0.289 | 0.000007 | 40285.71429 |
| 12/1/07 10:00 AM | 89.5 | 0.281 | 0.000007 | 39142.85714 |
| 12/3/07 8:00 AM | 135.5 | 0.287 | 0.000011 | 25090.90909 |
| 12/4/07 8:00 AM | 159.5 | 0.339 | 0.000012 | 27250 |
| 12/5/07 9:00 AM | 184.5 | 0.159 | 0.000001 | 158000 |
| 12/6/07 8:00 AM | 207.5 | 0.28 | 0.000005 | 55000 |
| average | | | | 42676.0241 |

First setup measured and calculated parameters

11/20/07

| R(ohm) | V(v) | I(A) | I(mA) | P(W) | P(mW) |
|---|---|---|---|---|---|
| 10 | 0 | 0 | 0 | 0 | 0 |
| 100 | 0.002 | 0.00002 | 0.02 | 0.00000004 | 0.00004 |
| 1000 | 0.023 | 0.000023 | 0.023 | 0.000000529 | 0.000529 |
| 10000 | 0.134 | 0.0000134 | 0.0134 | 1.7956E-06 | 0.001796 |
| 100000 | 0.296 | 0.00000296 | 0.00296 | 8.7616E-07 | 0.000876 |
| 1000000 | 0.35 | 0.00000035 | 0.00035 | 1.225E-07 | 0.000123 |

11/24/07

| R(ohm) | V(v) | I(A) | I(mA) | P(W) | P(mW) |
|---|---|---|---|---|---|
| 10 | 0 | 0 | 0 | 0 | 0 |
| 100 | 0.002 | 0.00002 | 0.02 | 0.00000004 | 0.00004 |
| 1000 | 0.018 | 0.000018 | 0.018 | 0.000000324 | 0.000324 |
| 10000 | 0.118 | 0.0000118 | 0.0118 | 1.3924E-06 | 0.0013924 |
| 100000 | 0.247 | 0.00000247 | 0.00247 | 6.1009E-07 | 0.00061009 |
| 1000000 | 0.277 | 0.000000277 | 0.000277 | 7.6729E-08 | 0.000076729 |

11/26/07

| R(ohm) | V(v) | I(A) | I(mA) | P(W) | P(mW) |
|---|---|---|---|---|---|
| 10 | 0 | 0 | 0 | 0 | 0 |
| 100 | 0.001 | 0.00001 | 0.01 | 0.00000001 | 0.00001 |
| 1000 | 0.014 | 0.000014 | 0.014 | 0.000000196 | 0.000196 |
| 10000 | 0.094 | 0.0000094 | 0.0094 | 8.836E-07 | 0.0008836 |
| 100000 | 0.215 | 0.00000215 | 0.00215 | 4.6225E-07 | 0.00046225 |
| 1000000 | 0.25 | 0.00000025 | 0.00025 | 6.25E-08 | 0.0000625 |

Second setup measured and calculated parameters

11/27/07

| R(ohm) | V(v) | I(A) | I(mA) | P(mW) |
|---|---|---|---|---|
| 10 | 0 | 0 | 0 | 0 |
| 100 | 0.004 | 0.00004 | 0.04 | 0.00016 |
| 1000 | 0.034 | 0.000034 | 0.034 | 0.001156 |
| 10000 | 0.08 | 0.000008 | 0.008 | 0.00064 |
| 100000 | 0.135 | 0.00000135 | 0.00135 | 0.00018225 |
| 1000000 | 0.15 | 0.00000015 | 0.00015 | 0.0000225 |

11/28/07 (data taken at 9 am)

| R(ohm) | V(v) | I(A) | I(mA) | P(mW) |
|---|---|---|---|---|
| 10 | 0 | 0 | 0 | 0 |
| 100 | 0.001 | 0.00001 | 0.01 | 0.00001 |
| 1000 | 0.011 | 0.000011 | 0.011 | 0.000121 |
| 10000 | 0.087 | 0.0000087 | 0.0087 | 0.0007569 |
| 100000 | 0.252 | 0.00000252 | 0.00252 | 0.00063504 |
| 1000000 | 0.311 | 0.000000311 | 0.000311 | 0.000096721 |

11/28/07 (data taken at 1.30 pm)

| R(ohm) | V(v) | I(A) | I(mA) | P(mW) |
|---|---|---|---|---|
| 10 | 0 | 0 | 0 | 0 |
| 100 | 0.002 | 0.00002 | 0.02 | 0.00004 |
| 1000 | 0.021 | 0.000021 | 0.021 | 0.000441 |
| 10000 | 0.134 | 0.0000134 | 0.0134 | 0.0017956 |
| 100000 | 0.28 | 0.0000028 | 0.0028 | 0.000784 |
| 1000000 | 0.316 | 0.000000316 | 0.000316 | 0.000099856 |

11/29/07 (data taken at 8 am)

| R(ohm) | V(v) | I(A) | I(mA) | P(mW) |
|---|---|---|---|---|
| 10 | 0 | 0 | 0 | 0 |
| 100 | 0.001 | 0.00001 | 0.01 | 0.00001 |
| 1000 | 0.008 | 0.000008 | 0.008 | 0.000064 |
| 10000 | 0.065 | 0.0000065 | 0.0065 | 0.0004225 |
| 100000 | 0.228 | 0.00000228 | 0.00228 | 0.00051984 |
| 1000000 | 0.303 | 0.000000303 | 0.000303 | 0.000091809 |

11/29/07 (data taken at 4.30 pm)

| R(ohm) | V(v) | I(A) | I(mA) | P(mW) |
|---|---|---|---|---|
| 10 | 0.0001 | 0.00001 | 0.01 | 0.000001 |
| 100 | 0.0008 | 0.000008 | 0.008 | 0.0000064 |
| 1000 | 0.0081 | 0.0000081 | 0.0081 | 0.00006561 |
| 10000 | 0.066 | 0.0000066 | 0.0066 | 0.0004356 |
| 100000 | 0.246 | 0.00000246 | 0.00246 | 0.00060516 |
| 1000000 | 0.342 | 0.000000342 | 0.000342 | 0.000116964 |

11/30/07

| R(ohm) | V(v) | I(A) | I(mA) | P(mW) |
|---|---|---|---|---|
| 10 | 0.0001 | 0.00001 | 0.01 | 0.000001 |
| 100 | 0.0007 | 0.000007 | 0.007 | 0.0000049 |
| 1000 | 0.007 | 0.000007 | 0.007 | 0.000049 |
| 10000 | 0.057 | 0.0000057 | 0.0057 | 0.0003249 |
| 100000 | 0.206 | 0.00000206 | 0.00206 | 0.00042436 |
| 1000000 | 0.277 | 0.000000277 | 0.000277 | 0.000076729 |

12/01/07

| R(ohm) | V(v) | I(A) | I(mA) | P(mW) |
|---|---|---|---|---|
| 10 | 0.0001 | 0.00001 | 0.01 | 0.000001 |
| 100 | 0.0007 | 0.000007 | 0.007 | 0.0000049 |
| 1000 | 0.007 | 0.000007 | 0.007 | 0.000049 |
| 10000 | 0.059 | 0.0000059 | 0.0059 | 0.0003481 |
| 100000 | 0.204 | 0.00000204 | 0.00204 | 0.00041616 |
| 1000000 | 0.273 | 0.000000273 | 0.000273 | 0.000074529 |

12/03/07

| R(ohm) | V(v) | I(A) | I(mA) | P(mW) |
|---|---|---|---|---|
| 10 | 0 | 0 | 0 | 0 |
| 100 | 0.001 | 0.00001 | 0.01 | 0.00001 |
| 1000 | 0.011 | 0.000011 | 0.011 | 0.000121 |
| 10000 | 0.082 | 0.0000082 | 0.0082 | 0.0006724 |
| 100000 | 0.235 | 0.00000235 | 0.00235 | 0.00055225 |
| 1000000 | 0.293 | 0.000000293 | 0.000293 | 0.000085849 |

12/04/07

| R(ohm) | V(v) | I(A) | I(mA) | P(mW) |
|---|---|---|---|---|
| 10 | 0.0001 | 0.00001 | 0.01 | 0.000001 |
| 100 | 0.001 | 0.00001 | 0.01 | 0.00001 |
| 1000 | 0.012 | 0.000012 | 0.012 | 0.000144 |
| 10000 | 0.093 | 0.0000093 | 0.0093 | 0.0008649 |
| 100000 | 0.268 | 0.00000268 | 0.00268 | 0.0007182 |
| 1000000 | 0.333 | 0.000000333 | 0.000333 | 0.0001109 |

12/05/07

| R(ohm) | V(v) | I(A) | I(mA) | P(mW) |
|---|---|---|---|---|
| 10 | 0 | 0 | 0 | 0 |
| 100 | 0.0002 | 0.000002 | 0.002 | 0.0000004 |
| 1000 | 0.001 | 0.000001 | 0.001 | 0.000001 |
| 10000 | 0.008 | 0.0000008 | 0.0008 | 0.0000064 |
| 100000 | 0.019 | 0.00000019 | 0.00019 | 0.00000361 |
| 1000000 | 0.021 | 0.000000021 | 0.000021 | 0.000000441 |

12/06/07

| R(ohm) | V(v) | I(A) | I(mA) | P(mW) |
|---|---|---|---|---|
| 10 | 0.0001 | 0.00001 | 0.01 | 0.000001 |
| 100 | 0.0005 | 0.000005 | 0.005 | 0.0000025 |
| 1000 | 0.005 | 0.000005 | 0.005 | 0.000025 |
| 10000 | 0.046 | 0.0000046 | 0.0046 | 0.0002116 |
| 100000 | 0.185 | 0.00000185 | 0.00185 | 0.00034225 |
| 1000000 | 0.267 | 0.000000267 | 0.000267 | 0.000071289 |

## 10.4. Appendix IV - Product Information Sheet of ATCC® 53774

**ATCC ® NUMBER: 53774**

**ORGANISM:** *Geobacter metallireducens*
Type strain GS-15. This material is cited in a U.S. and/or other Patent Application and may not be used to infringe the patent claims.

**CITATION STRAIN:**

If use of this strain results in a scientific publication it should be cited in that manuscript in the following manner: *Geobacter metallireducens* ATCC ® 53774.

**ATCC ® MEDIUM:**
#1768 Broth: Medium for Geobacter metallireducens

Milli Q H2O, 800mL
Ferric citrate (Sigma F-6129), 13.70 g
Vitamins (*see below*), 10mL
Minerals (*see below*), 10mL
$NaHCO_3$, 2.50g
$NH_4Cl$, 0.25g
$NaH_2PO4*H2O$, 0.60 g
KCl, 0.1 g
Sodium acetate, 6.8 g
*Heat about 400 ml of Milli Q $H_2O$ on a hot/stir plate to near boiling. Add ferric citrate, allow to dissolve then cool to room temperature in a slurry of ice, bring the volume of Milli Q $H_2O$ up to 800 ml by adding 400 ml Milli Q (this quickly cools the media to room temp). Adjust the pH to 6.0 using 10N NaOH (When the pH approaches 5.0 add the NaOH drop wise.). The approximate volume of 10N NaOH per liter is 6 ml.*

*Add the remaining ingredients and bring the final volume up to one liter with Milli Q $H_2O$. Bubble the medium with 80% $N_2$ and 20% $CO_2$. The final pH should be around 6.8-7.0.* **This medium should not be exposed to light.**

**CONDITIONS:**
Temperature: 30C
Atmosphere: Anaerobic (80% $N_2$, 20% $CO_2$)

**BIOSAFETY LEVEL: 1**

Appropriate safety procedures should always be used with this material. Laboratory safety is discussed in the following publication: *Biosafety in Microbiological and Biomedical Laboratories,* 4th ed. HHS Publication No. (CDC) 93-8395. U.S. Department of Health and Human Services, Centers for Disease Control and Prevention. Washington DC: U.S. Government Printing Office; 1999. The entire text is available online at www.cdc.gov/od/ohs/biosfty/bmb14/bmb14toc.htm .

## 10.5. Appendix V - Membrane Testing Data

Cellephane (1)

| | mL |
|---|---|
| NaOH | 1 |
| H2O | 29 |
| HCL (1M) | 100 |

Nafion

| | mL |
|---|---|
| NaOH | 1 |
| H2O | 29 |
| HCl (1M) | 100 |

| Time (min) | pH | [H] (mol/L) | [H]/L |
|---|---|---|---|
| 0 | 13.21 | 6.17E-14 | 3.71E+10 |
| 2 | 13.1 | 7.94E-14 | 4.78E+10 |
| 3 | 12.97 | 1.07E-13 | 6.45E+10 |
| 4 | 12.87 | 1.35E-13 | 8.12E+10 |
| 5 | 13.83 | 1.48E-14 | 8.90E+09 |
| 7 | 13.09 | 8.13E-14 | 4.89E+10 |
| 10 | 12.91 | 1.23E-13 | 7.41E+10 |
| 13 | 12.78 | 1.66E-13 | 9.99E+10 |
| 15 | 12.75 | 1.78E-13 | 1.07E+11 |
| 31 | 12.72 | 1.91E-13 | 1.15E+11 |
| 180 | 12.08 | 8.32E-13 | 5.01E+11 |
| 1320 | 10.94 | 1.15E-11 | 6.91E+12 |
| 1689 | 10.71 | 1.95E-11 | 1.17E+13 |
| 1740 | 10.7 | 2.00E-11 | 1.20E+13 |

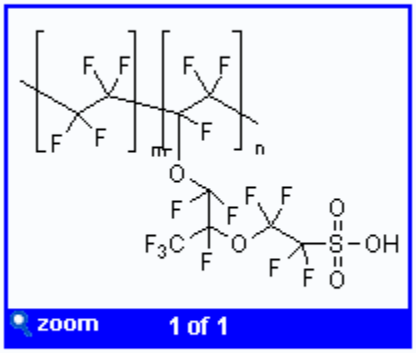
| Time (min) | pH | [H] (mol/L) | [H]/L |
|---|---|---|---|
| 0 | 13.35 | 4.47E-14 | 2.69E+10 |
| 1 | 13.31 | 4.90E-14 | 2.95E+10 |
| 2 | 13.13 | 7.41E-14 | 4.46E+10 |
| 3 | 13.09 | 8.13E-14 | 4.89E+10 |
| 10 | 12.85 | 1.41E-13 | 8.51E+10 |
| 36 | 12.32 | 4.79E-13 | 2.88E+11 |

## 10.6. Appendix VI - Nafion 117 Information sheet

# 274674 Nafion® perfluorinated membrane

Aldrich

**Nafion® 117, thickness 0.007 in.**




| | |
|---|---|
| **CAS Number** | 31175-20-9 |
| **MDL number** | MFCD00132790 |

Expand/Collapse All

### Price and Availability

| Product Number | Your Price USD | Available to Ship | Quantity | Actions |
|---|---|---|---|---|
| 274674-1EA | 297.00 | 04/23/2008 details... | | |

### Descriptions

**Application**

Nafion® is a perfluorinated ion-exchange membrane which has a wide variety of commercial uses. The acidic (hydrogen ion) form and derivatives are solid, superacid catalysts useful in a wide variety of synthetic applications.[1]

**Packaging**

1 ea in foil bag

As a proton exchange membrane Nafion® has been the focus of considerable efforts to develop improved fuel cells[2,3]

**Legal Information**

Nafion is a registered trademark of E.I. du Pont de Nemours & Co., Inc.

### Properties

| | |
|---|---|
| **eq. wt.** | 1,100 |
| **mfr. no.** | Nafion® 117 |
| **length × W** | 8 in. × 10 in. |
| **thickness** | 0.007 in. |
| **solubility** | $H_2O$: insoluble |

## 10.7. Appendix VII - Software Tools

AVR Butterfly kit has its own development tools for programming and debugging software called AVR Studio 4. It is available free of charge from the vendor. This program downloads the compiled code to the micro-controller. The following figure shows the IDE interface.

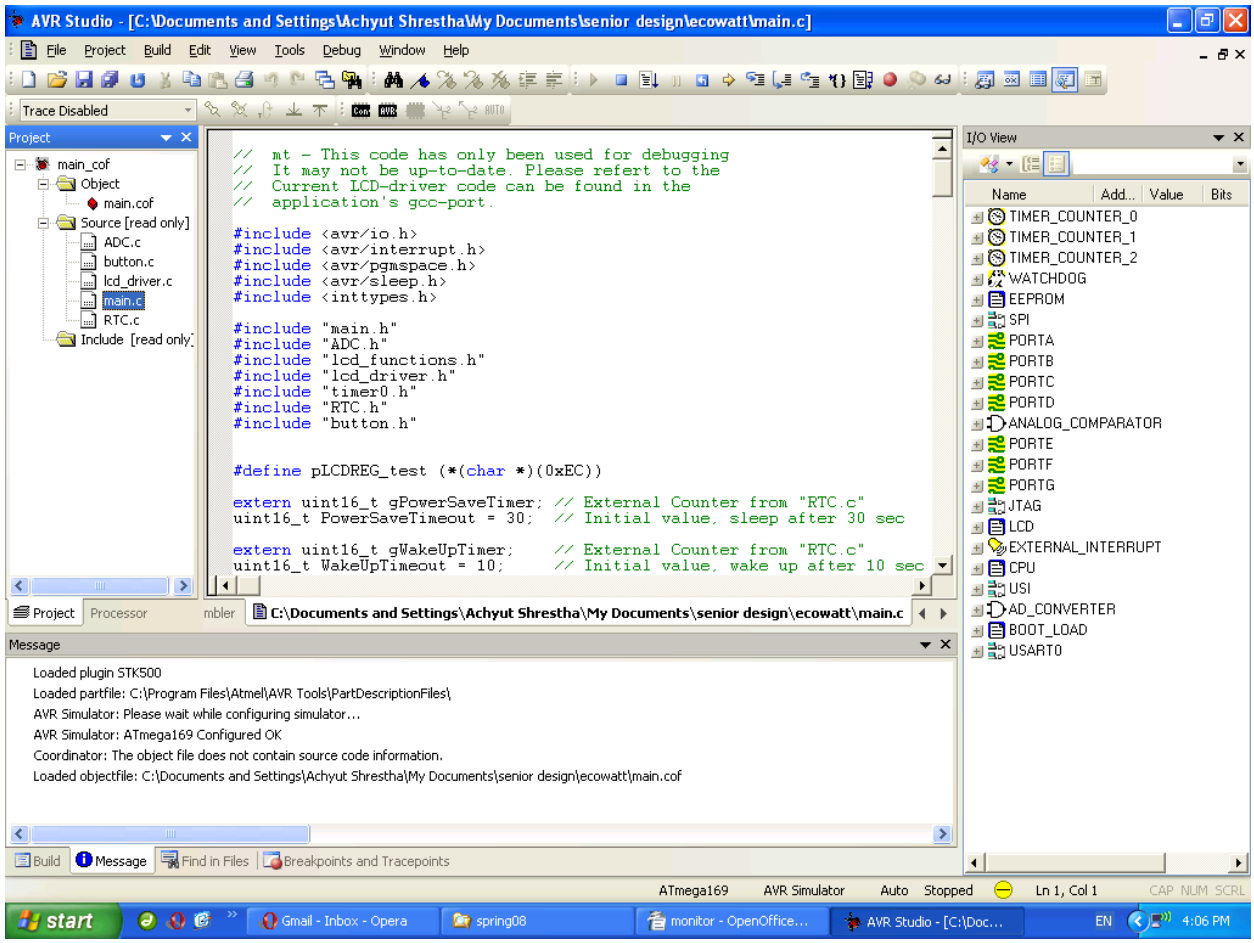



Figure 41 - AVR Studio 4 IDE

However, writing and compiling the program must happen before the tool accepts the importation of the program. For this, one uses an open source program called WinAVR (pronounced "whenever"). The code for the micro-controller is written in C with the help of this tool and then complied which will produce the hex file needed for downloading into the micro-controller with AVR studio 4. The following figure is the IDE layout of the program.

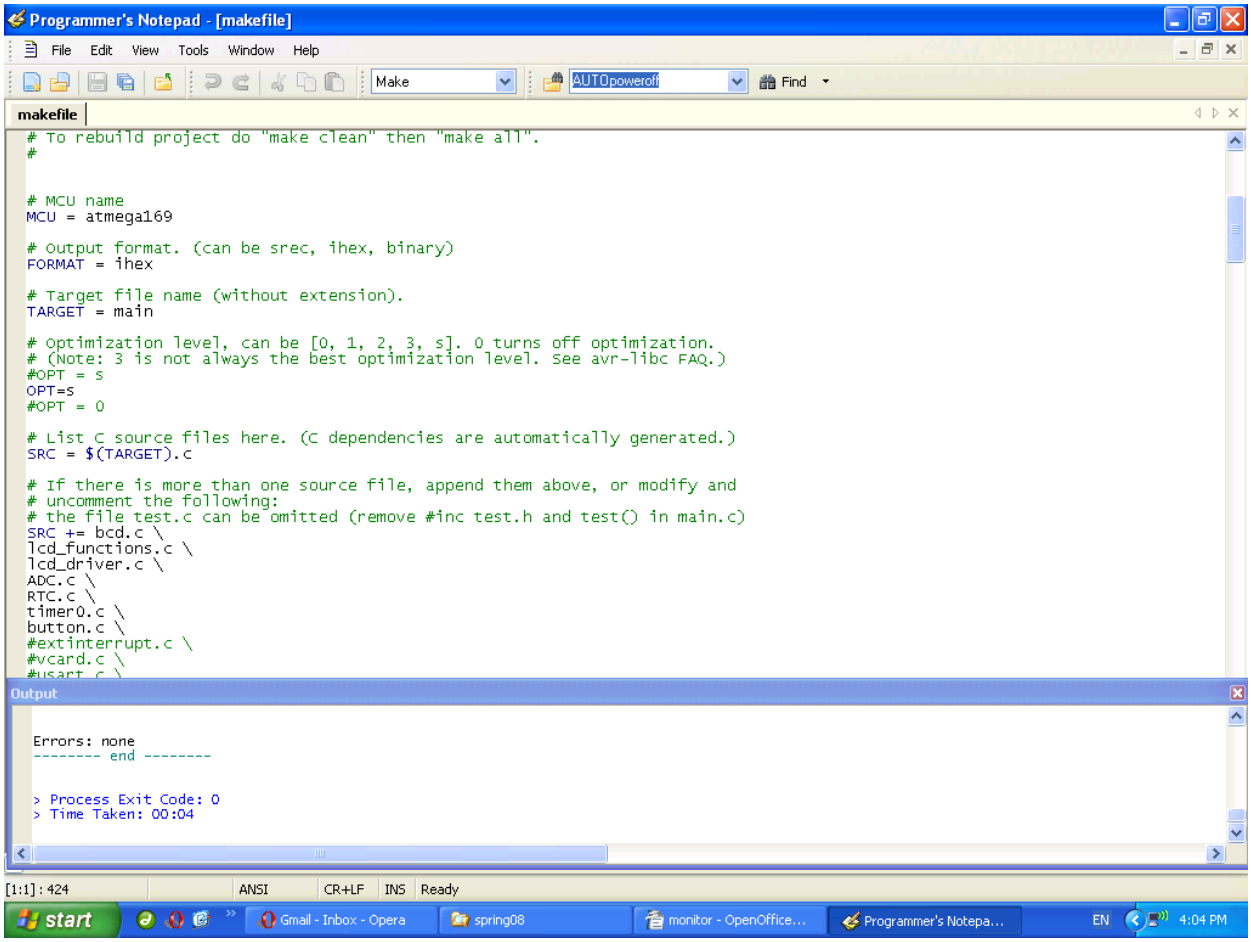



Figure 42 - WinAVR IDE

## 10.8. Appendix VIII – ATMega169 Specifications

### 10.8.1. Features

• High Performance, Low Power AVR® 8-Bit Microcontroller
• Advanced RISC Architecture
– 130 Powerful Instructions – Most Single Clock Cycle Execution
– 32 x 8 General Purpose Working Registers
– Fully Static Operation
– Up to 16 MIPS Throughput at 16 MHz
– On-Chip 2-cycle Multiplier
• Non-volatile Program and Data Memories
– 16K bytes of In-System Self-Programmable Flash
Endurance: 10,000 Write/Erase Cycles
– Optional Boot Code Section with Independent Lock Bits
In-System Programming by On-chip Boot Program
True Read-While-Write Operation
– 512 bytes EEPROM
Endurance: 100,000 Write/Erase Cycles
– 1K byte Internal SRAM
– Programming Lock for Software Security
• JTAG (IEEE std. 1149.1 compliant) Interface
– Boundary-scan Capabilities According to the JTAG Standard
– Extensive On-chip Debug Support
– Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
• Peripheral Features
– 4 x 25 Segment LCD Driver
– Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
– One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
– Real Time Counter with Separate Oscillator
– Four PWM Channels
– 8-channel, 10-bit ADC
– Programmable Serial USART
– Master/Slave SPI Serial Interface
– Universal Serial Interface with Start Condition Detector
– Programmable Watchdog Timer with Separate On-chip Oscillator
– On-chip Analog Comparator
– Interrupt and Wake-up on Pin Change
• Special Microcontroller Features
– Power-on Reset and Programmable Brown-out Detection
– Internal Calibrated Oscillator
– External and Internal Interrupt Sources
– Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby
• I/O and Packages
– 53 Programmable I/O Lines
– 64-lead TQFP and 64-pad QFN/MLF
• Speed Grade:
– ATmega169V: 0 - 4 MHz @ 1.8 - 5.5V, 0 - 8 MHz @ 2.7 - 5.5V
– ATmega169: 0 - 8 MHz @ 2.7 - 5.5V, 0 - 16 MHz @ 4.5 - 5.5V
• Temperature range:
– -40°C to 85°C Industrial
• Ultra-Low Power Consumption
– Active Mode:
1 MHz, 1.8V: 350µA
32 kHz, 1.8V: 20µA (including Oscillator)
32 kHz, 1.8V: 40µA (including Oscillator and LCD)
– Power-down Mode:
0.1µA at 1.8V

### 10.8.2. Schematic

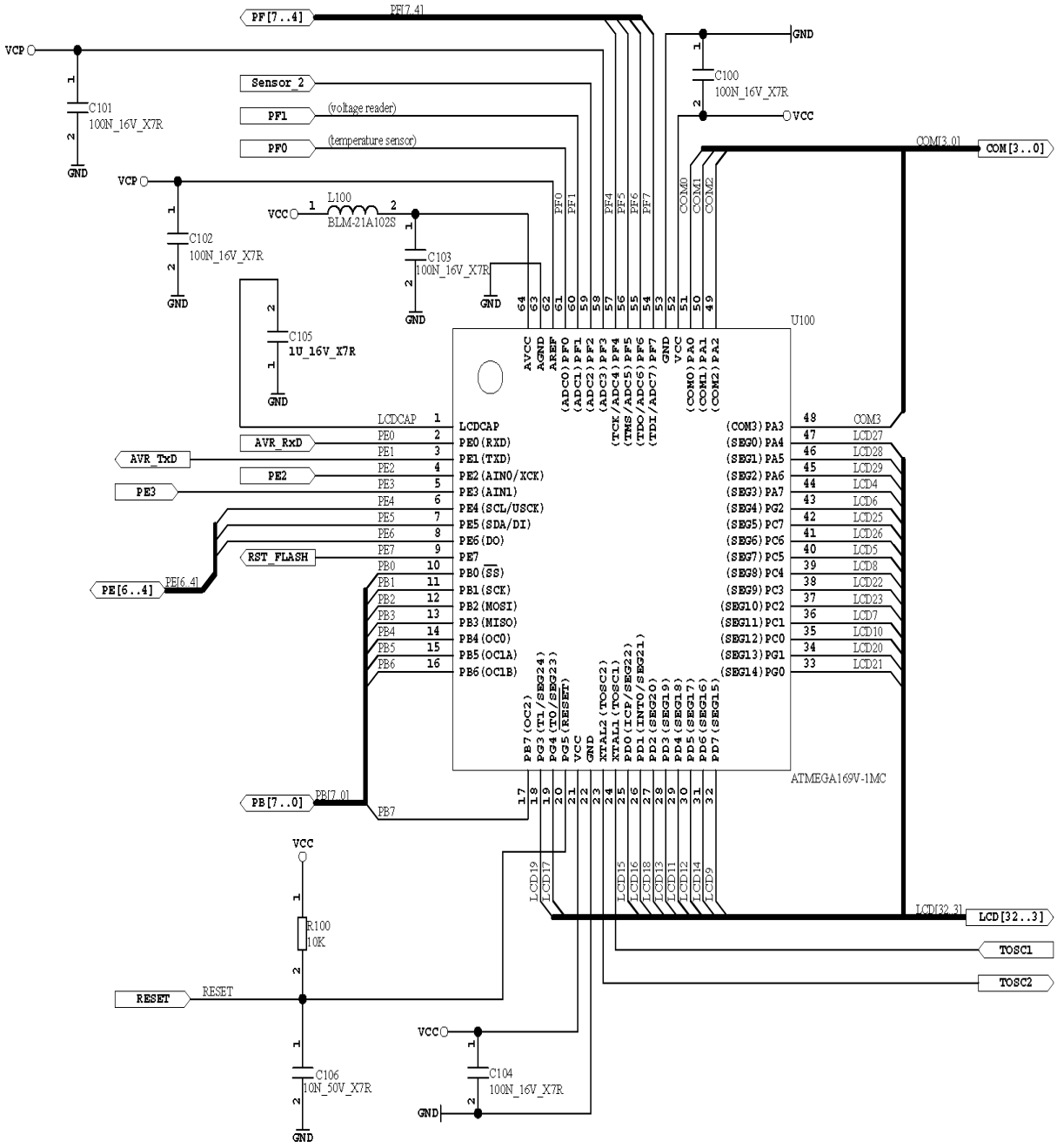



Figure 43 - ATMega169 Schematic

## 10.9. Appendix IX – Monitor Source Code (Written in C)

```
"main.c"
//************************************
************************************
//
//  File........: main.c
//
//  Author(s)...: Achyut Shrestha
//
//  Target(s)...: ATmega169
//
//   Compiler....: AVR-GCC 3.3.1; avr-
libc 1.2
//
//    Description.: AVR Butterfly main
routines
//
//  Revisions...: 1.0
//
//      YYYYMMDD   -   VER.   -   COMMENT
- SIGN.
//
// 20080503   main routine   - Achyut
Shrestha
//
//************************************
************************************

#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
#include <avr/sleep.h>
#include <inttypes.h>

#include "main.h"

#include "lcd_driver.h"
#include "timer0.h"
#include "RTC.h"



#define pLCDREG_test (*(char *)(0xEC))

extern   uint16_t   gPowerSaveTimer;   //
External Counter from "RTC.c"
uint16_t  PowerSaveTimeout  =  30;    //
Initial value, sleep after 30 sec

extern  uint16_t  gWakeUpTimer;       //
External Counter from "RTC.c"
uint16_t  WakeUpTimeout  =  10;       //
Initial value, wake up after 10 sec


const  int  TEMP_Celcius_pos[]  PROGMEM  =
// Positive Celcius temperatures (ADC-
value)
{                              // from 0
to 60 degrees
      806,796,786,775,765,754,743,732,7
20,709,697,685,673,661,649,
      636,624,611,599,586,574,562,549,5
37,524,512,500,488,476,464,
      452,440,429,418,406,396,385,374,3
64,354,344,334,324,315,306,
      297,288,279,271,263,255,247,240,2
33,225,219,212,205,199,193,
      187,
};


const  int  TEMP_Celcius_neg[]  PROGMEM  =
// Negative Celcius temperatures (ADC-
value)
{                              // from -1
to -15 degrees
      815,825,834,843,851,860,868,876,8
83,891,898,904,911,917,923,
};

char   degree = CELCIUS; // char degree
= CELCIUS;


#define FALSE   0
#define TRUE    (!FALSE)

char CONTRAST = LCD_INITIAL_CONTRAST;

//  Start-up  delay  before  scrolling  a
string over the LCD. "LCD_driver.c"
extern char gLCD_Start_Scroll_Timer;


#define BOOL    char
#define FALSE   0
#define TRUE    (!FALSE)

// Variable from "button.c" to prevent
button-bouncing
extern unsigned char gButtonTimeout;

extern BOOL gAutoPressJoystick;

//  Used  to  indicate  when  the  LCD
interrupt handler should update the LCD
//  mt  jw  char  gLCD_Update_Required  =
FALSE;
volatile  char  gLCD_Update_Required  =
FALSE;

//  LCD  display  buffer  (for  double
buffering).
char LCD_Data[LCD_REGISTER_COUNT];

// Buffer that contains the text to be
displayed
//  Note:  Bit  7  indicates  that  this
character is flashing
char gTextBuffer[TEXTBUFFER_SIZE];
```

```
// Only six letters can be shown on the
LCD.
// With the gScroll and gScrollMode
variables,
// one can select which part of the
buffer to show
volatile signed char gScroll;
volatile char gScrollMode;

////Start-up delay before scrolling a
string over the LCD
char gLCD_Start_Scroll_Timer = 0;

// The gFlashTimer is used to determine
the on/off
// timing of flashing characters
char gFlashTimer = 0;

// Turns on/off the colons on the LCD
char gColon = 0;


// Look-up table used when converting
ASCII to
// LCD display data (segment control)
//    mt    __flash    unsigned    int
LCD_character_table[] =
unsigned    int    LCD_character_table[]
PROGMEM =
{
    0x0A51,     // '*' (?)
    0x2A80,     // '+'
    0x0000,     // ',' (Not defined)
    0x0A00,     // '-'
    0x0A51,     // '.' Degree sign
    0x0000,     // '/' (Not defined)
    0x5559,     // '0'
    0x0118,     // '1'
    0x1e11,     // '2
    0x1b11,     // '3
    0x0b50,     // '4
    0x1b41,     // '5
    0x1f41,     // '6
    0x0111,     // '7
    0x1f51,     // '8
    0x1b51,     // '9'
    0x0000,     // ':' (Not defined)
    0x0000,     // ';' (Not defined)
    0x0000,     // '<' (Not defined)
    0x0000,     // '=' (Not defined)
    0x0000,     // '>' (Not defined)
    0x0000,     // '?' (Not defined)
    0x0000,     // '@' (Not defined)
    0x0f51,     // 'A' (+ 'a')
    0x3991,     // 'B' (+ 'b')
    0x1441,     // 'C' (+ 'c')
    0x3191,     // 'D' (+ 'd')
    0x1e41,     // 'E' (+ 'e')
    0x0e41,     // 'F' (+ 'f')
    0x1d41,     // 'G' (+ 'g')
    0x0f50,     // 'H' (+ 'h')
    0x2080,     // 'I' (+ 'i')
    0x1510,     // 'J' (+ 'j')
    0x8648,     // 'K' (+ 'k')
    0x1440,     // 'L' (+ 'l')
    0x0578,     // 'M' (+ 'm')
    0x8570,     // 'N' (+ 'n')
    0x1551,     // 'O' (+ 'o')
    0x0e51,     // 'P' (+ 'p')
    0x9551,     // 'Q' (+ 'q')
    0x8e51,     // 'R' (+ 'r')
    0x9021,     // 'S' (+ 's')
    0x2081,     // 'T' (+ 't')
    0x1550,     // 'U' (+ 'u')
    0x4448,     // 'V' (+ 'v')
    0xc550,     // 'W' (+ 'w')
    0xc028,     // 'X' (+ 'x')
    0x2028,     // 'Y' (+ 'y')
    0x5009,     // 'Z' (+ 'z')
    0x0000,     // '[' (Not defined)
    0x0000,     // '\' (Not defined)
    0x0000,     // ']' (Not defined)
    0x0000,     // '^' (Not defined)
    0x0000      // '_'
};

uint8_t gSECOND;
uint8_t gMINUTE;
uint8_t gHOUR;
uint8_t gDAY;
uint8_t gMONTH;
// mtE
unsigned int gYEAR;
// mtA
//char gPowerSaveTimer = 0;
//char dateformat = 0;
uint16_t gPowerSaveTimer = 0;
uint16_t gWakeUpTimer = 0;
uint8_t dateformat = 0;
// mtE

// Lookup table holding the length of
each month. The first element is a
dummy.
// mt this could be placed in progmem
too, but the arrays are accessed quite
//    often - so leaving them in RAM is
better...
char MonthLength[13] = {0, 31, 28, 31,
30, 31, 30, 31, 31, 30, 31, 30, 31};

char TBL_CLOCK_12[] =    // table used
when displaying 12H clock
{12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11};

char clockformat = CLOCK_12;     // set
initial clock format to 12H

// different date formates (text only)
// mtA
//__flash  char  EUROPEAN_DATE_TEXT[]  =
"DDMMYY";
//__flash  char  AMERICAN_DATE_TEXT[]  =
"MMDDYY";
```

```
//__flash  char  CANADIAN_DATE_TEXT[]  =
"YYMMDD";
const char EUROPEAN_DATE_TEXT[] PROGMEM
=   "DDMMYY";
const char AMERICAN_DATE_TEXT[] PROGMEM
=   "MMDDYY";
const char CANADIAN_DATE_TEXT[] PROGMEM
=   "YYMMDD";
// mtE

// different date formates, table for
putting DD, MM and YY at the right
place
// on the LCD
const     uint8_t     EUROPEAN_DATE_NR[]
PROGMEM =   { 4, 5, 2, 3, 0, 1 };
const     uint8_t     AMERICAN_DATE_NR[]
PROGMEM =   { 4, 5, 0, 1, 2, 3 };
const     uint8_t     CANADIAN_DATE_NR[]
PROGMEM =   { 0, 1, 2, 3, 4, 5 };

PGM_P         DATEFORMAT_TEXT[]         =
{EUROPEAN_DATE_TEXT,
AMERICAN_DATE_TEXT,
CANADIAN_DATE_TEXT};

TIMER_CALLBACK_FUNC
CallbackFunc[TIMER0_NUM_CALLBACKS];

uint8_t
CountDownTimers[TIMER0_NUM_COUNTDOWNTIM
ERS];


/*************************************
*************************************
*
*   Function name : main
*
*   Returns :       None
*
*   Parameters :    None
*
*    Purpose :         Contains the main
loop of the program
*
*************************************
*************************************/
int main(void)
{

      // Program initalization

    Initialization();
    sei(); // mt __enable_interrupt();

      char i;
      char input;

      for (;;){

            LCD_puts("ECOWATT",1);
            Delay(3000);

            if (!(PINE & 0x10))
            {

      LCD_UpdateRequired(TRUE,0);
                  for (;;)
                  {
                        if
(gPowerSaveTimer   >=   PowerSaveTimeout)
break;

      LCD_puts("Circuit error",1);
                        Delay(5000);
                  }
            }

            LCD_Clear();

      LCD_UpdateRequired(TRUE,0);
            char temp_err = 'F';

            int temp_val ;
            ADC_init(0);
            temp_val = ADC_read();
            ADC2Temp(temp_val);
            Delay(3000);
            LCD_Clear();

      LCD_UpdateRequired(TRUE,0);

            for (;;)
            {
                  temp_err          =
Temp_check();
                  if (temp_err == 'T'
) {
                        if
(gPowerSaveTimer   >=   PowerSaveTimeout)
break;

      LCD_puts("Temp out of range",1);
                        Delay(5000);
                  } else {
                        break;
                  }
            }

            LCD_puts("Optimum
Temp",1);
            Delay(5000);

            LCD_Clear();

      LCD_UpdateRequired(TRUE,0);
            for(;;)
            {
                  Power_check();
                  Delay(4000);
                  if  (gPowerSaveTimer
>= PowerSaveTimeout) break;
            }

            gPowerSaveTimer = 0;
```

```
            LCD_puts_f(PSTR("bye"),1);
            Delay(3000);
            SMCR = (2<<SM0) | (1<<SE);
// Enable Power-down mode
            asm volatile ("sleep"::);

            if(!(PINB     &     0x40))
// press UP to wake from SLEEP
        {
            //PowerSave = FALSE;
            for(i = 0; i < 20; i++) //
set all LCD segment register to the
variable ucSegments
            {
                *(&pLCDREG_test + i) =
0x00;
            }

            sbi(LCDCRA,              7);
// enable LCD
            input      =      getkey();
// Read buttons
                    gPowerSaveTimer    =
0;       // reset seconds until sleep
timer
        }
            SMCR          =          0;
// Just woke, disable sleep

      } //End Main loop

      return 0;
}

/*************************************
*************************************
*
*   Function name : Initialization
*
*   Returns :       None
*
*   Parameters :    None
*
*    Purpose :          Initializate the
different modules
*
*************************************
*************************************/
void Initialization(void)
{
      OSCCAL_calibration();          //
calibrate the OSCCAL byte

      CLKPR = (1<<CLKPCE);           //
set Clock Prescaler Change Enable

    // set prescaler = 8, Inter RC 8Mhz
/ 8 = 1Mhz
    CLKPR = (1<<CLKPS1) | (1<<CLKPS0);

    // Disable Analog Comparator (power
save)
    ACSR = (1<<ACD);

    // Disable Digital input on PF0-2
(power save)
    DIDR1 = (7<<ADC0D);

    //   mt   PORTB   =   (15<<PORTB0);
// Enable pullup on
      PORTB = (15<<PB0);             //
Enable pullup on
    // mt PORTE = (15<<PORTE4);
      PORTE = (15<<PE4);

    sbi(DDRB, 5);                  // set
OC1A as output
    sbi(PORTB, 5);                 // set
OC1A high

      Button_Init();                 //
Initialize   pin   change   interrupt   on
joystick


      RTC_init();                    //
Start timer2 asynchronous, used for RTC
clock

      Timer0_Init();             //
setup timers for debounce and scroll

    LCD_Init();                        //
initialize the LCD

}


/*************************************
*************************************
*
*   Function name : OSCCAL_calibration
*
*   Returns :       None
*
*   Parameters :    None
*
*    Purpose :           Calibrate the
internal   OSCCAL   byte,   using   the
external
*                     32,768 kHz crystal
as reference
*
*************************************
*************************************/
void OSCCAL_calibration(void)
{
    unsigned char calibrate = FALSE;
    int temp;
    unsigned char tempL;

    CLKPR = (1<<CLKPCE);          // set
Clock Prescaler Change Enable

            //   set   prescaler   =   8,
Inter RC 8Mhz / 8 = 1Mhz
```

```
    CLKPR = (1<<CLKPS1) | (1<<CLKPS0);

    TIMSK2 = 0;                //disable
OCIE2A and TOIE2

    ASSR = (1<<AS2);           //select
asynchronous    operation    of    timer2
(32,768kHz)

    OCR2A = 200;                // set
timer2 compare value

    TIMSK0 = 0;                // delete
any interrupt sources

    TCCR1B = (1<<CS10);        // start
timer1 with no prescaling
    TCCR2A = (1<<CS20);        // start
timer2 with no prescaling

    while((ASSR   &   0x01)   |   (ASSR   &
0x04));           //wait  for  TCN2UB  and
TCR2UB to be cleared

    Delay(1000);          //   wait   for
external crystal to stabilise

    while(!calibrate)
    {
        cli();           //           mt
__disable_interrupt();      //   disable
global interrupt

        TIFR1 = 0xFF;    // delete TIFR1
flags
        TIFR2 = 0xFF;    // delete TIFR2
flags

        TCNT1H = 0;      // clear timer1
counter
        TCNT1L = 0;
        TCNT2 = 0;       // clear timer2
counter

        while  (  !(TIFR2  &&  (1<<OCF2A))
);   // wait for timer2 compareflag

        TCCR1B = 0; // stop timer1

        sei(); // __enable_interrupt();
// enable global interrupt

        if ( (TIFR1 && (1<<TOV1)) )
        {
            temp = 0xFFFF;        // if
timer1   overflows,   set   the   temp   to
0xFFFF
        }
        else
        {      //  read  out  the  timer1
counter value
            tempL = TCNT1L;
            temp = TCNT1H;
            temp = (temp << 8);
            temp += tempL;
        }

        if (temp > 6250)
        {
            OSCCAL--;    // the internRC
oscillator  runs  to  fast,  decrease  the
OSCCAL
        }
        else if (temp < 6120)
        {
            OSCCAL++;    // the internRC
oscillator  runs  too  slow,  increase  the
OSCCAL
        }
        else
            calibrate = TRUE;    // the
interRC is correct

        TCCR1B  =  (1<<CS10);  //  start
timer1
    }
}


/*************************************
*************************************
*
*   Function name : Delay
*
*   Returns :       None
*
*     Parameters  :        unsigned  int
millisec
*
*    Purpose :         Delay-loop  (based
on a 1Mhz Clock)
*
*************************************
*************************************/
void Delay(unsigned int millisec)
{
    // mt, int i did not work in the
simulator:  int i;
       uint8_t i;

    while (millisec--)
              //mt:   for   (i=0;   i<125;
i++);
              for (i=0; i<125; i++)
                    asm         volatile
("nop"::);
}



/*************************************
*************************************
*
*   Function name : ADC_init
*
```

```
*   Returns :       None
*
*   Parameters :    char input
*
*    Purpose :        Initialize the ADC
with the selected ADC-channel
*
************************************
************************************/
void ADC_init(char input)
{
       ADMUX  =  input;       //  external
AREF and ADCx

    ADCSRA  =  (1<<ADEN)  |  (1<<ADPS1)  |
(1<<ADPS0);     // set ADC prescaler to
, 1MHz / 8 = 125kHz
       input  =  ADC_read();           //
dummy
}


/************************************
************************************
*
*   Function name : ADC_read
*
*   Returns :       int ADC
*
*   Parameters :    None
*
*    Purpose :          Do  a  Analog  to
Digital Conversion
*
************************************
************************************/
int ADC_read(void)
{
    char i;
    int ADC_temp;
    int ADCr = 0;

    // To save power, the voltage over
the LDR and the NTC is turned off when
not used
    // This is done by controlling the
voltage from a I/O-pin (PORTF3)

    sbi(PORTF,  PORTF3);       // Enable
the VCP (VC-peripheral)
    sbi(DDRF, PORTF3);

    sbi(ADCSRA,  ADEN);        //  Enable
the ADC

    //do a dummy readout first
    ADCSRA  |=  (1<<ADSC);          // do
single conversion
    while(!(ADCSRA & 0x10));     // wait
for conversion done, ADIF flag active
    for(i=0;i<8;i++)               // do
the ADC conversion 8 times for better
accuracy
    {
        ADCSRA |= (1<<ADSC);          //
do single conversion
        while(!(ADCSRA & 0x10));      //
wait  for  conversion  done,  ADIF  flag
active

        ADC_temp = ADCL;              //
read out ADCL register
        ADC_temp += (ADCH << 8);      //
read out ADCH register

        ADCr  +=  ADC_temp;           //
accumulate result (8 samples) for later
averaging
    }

    ADCr = ADCr >> 3;       // average
the 8 samples

    cbi(PORTF,  PORTF3);       // disable
the VCP
    cbi(DDRF, PORTF3);

    cbi(ADCSRA, ADEN);        // disable
the ADC

    return ADCr;
}



/************************************
************************************
*
*   Function name : ADC2Temp
*
*   Returns :       nothing
*
*    Parameters :     char log (char to
log to rs232 else print to screen)
*                                int
ADCResult (ADC reading to convert)
*
*    Purpose :          Convert  an  ADC
reading into a temp
*
************************************
************************************/
void ADC2Temp(int ADCresult)
{
    //   int ADCresult = 0;
    //   int ADCresult_temp = 0;
    int Temp_int=0;
    int dif;
    int top;

    char Temp;
    unsigned char i = 0;
    char TL;
```

```
    char TH;
    char VH;
    char VL;

       if(ADCresult > 810){           //
If it's a negtive temperature
                    for    (i=0;    i<=25;
i++){   // Find the temperature
                            if (ADCresult
<=
pgm_read_word(&TEMP_Celcius_neg[i])){
                                    //  add
a    decimal    reading    through    liner
interpolation
                                    if
(ADCresult                                  <
pgm_read_word(&TEMP_Celcius_neg[i])){

       dif=
pgm_read_word(&TEMP_Celcius_neg[i])-
pgm_read_word(&TEMP_Celcius_neg[i-1]);

       top                    =(ADCresult-
pgm_read_word(TEMP_Celcius_neg[i-
1]))*100;


       Temp_int = top/(dif+1);

       i++;
                                    }
                                    break;
                            }
                    }

                    LCD_putc(0,    '-');
//  Put  a  minus  sign  in  front  of  the
temperature
             }
             else if (ADCresult < 800){
// If it's a positive temperature
                    for    (i=0;    i<60;
i++)  {
                            if (ADCresult
>=
pgm_read_word(&TEMP_Celcius_pos[i])){
                                    //  add
a    decimal    reading    through    liner
interpolation
                                    if
(ADCresult                                  >
pgm_read_word(&TEMP_Celcius_pos[i])){

       dif=
pgm_read_word(&TEMP_Celcius_pos[i-1])-
pgm_read_word(&TEMP_Celcius_pos[i]);

       top
=(pgm_read_word(&TEMP_Celcius_pos[i-
1])-ADCresult)*100;


       Temp_int = top/(dif+1);

       i--;
                                    }
                                    break;
                            }
                    }

             }                      else{
//If the temperature is zero degrees
                    i = 0;
                    LCD_putc(0, ' ');
             }

             Temp     =     CHAR2BCD2(i);
// Convert from char to bin
             TL = (Temp & 0x0F) + '0';
// Find the low-byte
             TH = (Temp >> 4) + '0';
// Find the high-byte

             Temp= CHAR2BCD2(Temp_int);
             VH = (Temp >> 4) + '0';
             VL = (Temp & 0x0F) + '0';

           LCD_putc(0, ' ');
           //LCD    character    1    is
allready written to
           LCD_putc(2, TH);
           LCD_putc(3, TL);
           LCD_putc(4, '*');
           LCD_putc(5, 'C');
           LCD_putc(6, '\0');

}


/*************************************
*************************************
*
*   Function name : ADC2Volt
*
*   Returns :       nothing
*
*    Parameters :     char log (char to
log to flash else print to screen)
*                                   int
ADCResult (ADC reading to convert)
*
*    Purpose :          Convert an ADC
reading into a Voltage
*
*************************************
*************************************/
void ADC2Volt(int ADCresult)
{

    // int ADCresult_temp = 0;
    char Temp;

    char TL;
    char VH;
    char VL;
```

```
    float V_ADC;
    char VoltageHB;
    char VoltageLB;

    V_ADC = ( ADCresult * VREF ) /
1024; // Calculate the voltage

    V_ADC = ( V_ADC * 6 );          //
Multiply by 6 because of the voltage
division

    VoltageHB = V_ADC;              //
Store the high-byte
    V_ADC = ( V_ADC - VoltageHB );
    VoltageLB = ( V_ADC * 100 );    //
Store the low-byte

    Temp = CHAR2BCD2(VoltageHB);    //
Convert from char to bin

    TL = (Temp & 0x0F) + '0';

    Temp = CHAR2BCD2(VoltageLB);    //
Convert from char to bin

    VH = (Temp >> 4) + '0';
    VL = (Temp & 0x0F) + '0';

       LCD_putc(0, ' ');
       LCD_putc(1, ' ');
       LCD_putc(2, TL);
       LCD_putc(3, 'v');
       LCD_putc(4, VH);
       LCD_putc(5, VL);
       LCD_putc(6, '\0');

}

/*****************************
*
*   Function name : Power_check
*
*   Returns :       nothing
*
*   Parameters :    none
*
*   Purpose :       display power level
*
*****************************/
void Power_check(void){
       int voltage = 0;

       ADC_init(1);
       voltage = ADC_read();

       PGM_P statetext;

       if  (voltage  >  PWR_MAX_V){  //
battery is really really good
              statetext   =   PSTR("Batt
Full");

       }else if (voltage> PWR_GOOD_V){
// battery is freshly charged
              statetext   =   PSTR("Batt
Good");

       }else if (voltage>PWR_LOW_V){ //
normal operation mode
              statetext   =   PSTR("Batt
OK");

       }else if (voltage>PWR_MIN_V){ //
only 12% left
              statetext   =   PSTR("Batt
Low");
       }else {
              statetext   =   PSTR("Batt
Flat");
       }
       LCD_UpdateRequired(TRUE,0);
       LCD_puts_f(statetext, 1);

       statetext = NULL;
}

char Temp_check(void){
       int temperature = 0;
       char temp_error = 'F';

       ADC_init(0);
       temperature = ADC_read();

       if   (temperature   >   574   ||
temperature < 396) {
              temp_error = 'T';
       } else {
              temp_error = 'F';
       }
       return temp_error;
}


/*************************************
*************************************
*
*   Function name : Button_Init
*
*   Returns :       None
*
*   Parameters :    None
*
*    Purpose :          Initializes the
five button pin
*
*************************************
*************************************/
void Button_Init(void)
{
    // Init port pins
    cbi(DDRB,7);
    cbi(DDRB,6);
    cbi(DDRB,4);
    PORTB |= PINB_MASK;
    DDRE = 0x00;
```

```
    PORTE |= PINE_MASK;

    // Enable pin change interrupt on
PORTB and PORTE
       PCMSK0 = PINE_MASK;
       PCMSK1 = PINB_MASK;
       EIFR = (1<<PCIF0)|(1<<PCIF1);
       EIMSK = (1<<PCIE0)|(1<<PCIE1);

       CountdownTimerHandle            =
Timer0_AllocateCountdownTimer();
}

SIGNAL(SIG_PIN_CHANGE0)
// mtE
{
       PinChangeInterrupt();

}


SIGNAL(SIG_PIN_CHANGE1)
{
    PinChangeInterrupt();
}


/*************************************
*************************************
*
*   Function name : PinChangeInterrupt
*
*   Returns :       None
*
*   Parameters :    None
*
*    Purpose :       Check status on the
joystick
*
*************************************
*************************************/
void PinChangeInterrupt(void)
{
    char buttons;

    char key;

/*
    Read the buttons:

    Bit              7    6    5    4    3
2   1   0
    ------------------------------------
----------
    PORTB           B    A       O
    PORTE                               D
C
    ------------------------------------
----------
    PORTB | PORTE    B    A        O    D
C
=====================================
======
*/


    buttons = (~PINB) & PINB_MASK;
    buttons |= (~PINE) & PINE_MASK;

    // Output virtual keys
    if (buttons & (1<<BUTTON_A))
        key = KEY_PLUS;
    else if (buttons & (1<<BUTTON_B))
        key = KEY_MINUS;
    else if (buttons & (1<<BUTTON_C))
        key = KEY_PREV;
    else if (buttons & (1<<BUTTON_D))
        key = KEY_NEXT;
    else if (buttons & (1<<BUTTON_O))
        key = KEY_ENTER;
    else
        key = KEY_NULL;


    if(key != KEY_NULL)
    {
        if(gButtonTimeout)            //
gButtonTimeout    is    set    in    the
LCD_SOF_interrupt in LCD_driver.c
        {
            if (!KEY_VALID)
            {
                KEY = key;            //
Store key in global key buffer
                KEY_VALID = TRUE;
            }

         gButtonTimeout = FALSE;

        }
    }

    EIFR  =  (1<<PCIF1)  |  (1<<PCIF0);
// Delete pin change interrupt flags

    gPowerSaveTimer         =         0;
// Reset the Auto Power Down timer

}


/*************************************
*************************************
*
*   Function name : getkey
*
*   Returns :       The valid key
*
*   Parameters :    None
*
*   Purpose :       Get the valid key
*
```

```
*************************************
*************************************/
char getkey(void)
{
    char k;

    cli();              //              mt:
__disable_interrupt();

    if (KEY_VALID)                      //
Check for unread key in buffer
    {
        k = KEY;
        KEY_VALID = FALSE;
    }
    else
        k = KEY_NULL;               // No
key stroke available

    sei(); // mt: __enable_interrupt();

    return k;
}


/*************************************
*************************************
*
*      Function name : LCD_puts_f
*
*      Returns :           None
*
*      Parameters : pFlashStr:   Pointer
to the string in flash
*                   scrollmode: Not in
use
*
*      Purpose :           Writes      a
string stored in flash to the LCD
*
*************************************
*************************************/

//   mt   void   LCD_puts_f(char   __flash
*pFlashStr, char scrollmode)
void  LCD_puts_f(const  char  *pFlashStr,
char scrollmode)
{
    // char i;
       uint8_t i;

    while        (gLCD_Update_Required);
// Wait for access to buffer

    // mt: for (i = 0; pFlashStr[i] &&
i < TEXTBUFFER_SIZE; i++)
       for         (i         =         0;
pgm_read_byte(&pFlashStr[i])   &&   i   <
TEXTBUFFER_SIZE; i++)
    {
        //    mt:    gTextBuffer[i]    =
pFlashStr[i];
            gTextBuffer[i]            =
pgm_read_byte(&pFlashStr[i]);
    }

    gTextBuffer[i] = '\0';

    if (i > 6)
    {
        gScrollMode  =  1;             //
Scroll  if  text  is  longer  than  display
size
        gScroll = 0;
        gLCD_Start_Scroll_Timer   =   3;
//Start-up  delay  before  scrolling  the
text
    }
    else
    {
        gScrollMode = 0;
        gScroll = 0;
    }

    gLCD_Update_Required = 1;
}


/*************************************
*************************************
*
*      Function name : LCD_puts
*
*      Returns :           None
*
*      Parameters : pStr:   Pointer   to
the string
*                   scrollmode: Not in
use
*
*      Purpose :           Writes      a
string to the LCD
*
*************************************
*************************************/
void    LCD_puts(char    *pStr,    char
scrollmode)
{
       uint8_t i; // char i;

       while     (gLCD_Update_Required);
// Wait for access to buffer

    for   (i   =   0;   pStr[i]   &&   i   <
TEXTBUFFER_SIZE; i++)
    {
        gTextBuffer[i] = pStr[i];
    }

    gTextBuffer[i] = '\0';

    if (i > 6)
    {
```

```
        gScrollMode = 1;            //
Scroll if text is longer than display
size
        gScroll = 0;
        gLCD_Start_Scroll_Timer   =   3;
//Start-up delay before scrolling the
text
    }
    else
    {
        gScrollMode = 0;
        gScroll = 0;
    }

    gLCD_Update_Required = 1;
}



/*************************************
*************************************
*   Function name : CHAR2BCD2
*     Returns :           Binary coded
decimal value of the input (2 digits)
*    Parameters :     Value between (0-
99) to be encoded into BCD
*   Purpose :       Convert a character
into a BCD encoded character.
*                     The input must be
in the range 0 to 99.
*                    The result is byte
where the high and low nibbles
*                      contain the tens
and ones of the input.
*************************************
*************************************/
char CHAR2BCD2(char input)
{
    char high = 0;


    while       (input       >=       10)
// Count tens
    {
        high++;
        input -= 10;
    }

    return     (high   <<   4)   |   input;
// Add ones and return answer
}

/*************************************
*************************************
*
*   Function name : CHAR2BCD3
*
*     Returns :           Binary coded
decimal value of the input (3 digits)
*
*    Parameters :     Value between (0-
255) to be encoded into BCD
*
*   Purpose :       Convert a character
into a BCD encoded character.
*                     The input must be
in the range 0 to 255.
*                      The result is an
integer where the three lowest nibbles
*                     contain the ones,
tens and hundreds of the input.
*
*************************************
*************************************/
unsigned int CHAR2BCD3(char input)
{
    int high = 0;

    while      (input      >=      100)
// Count hundreds
    {
        high++;
        input -= 100;
    }

    high <<= 4;

    while       (input       >=       10)
// Count tens
    {
        high++;
        input -= 10;
    }

    return     (high   <<   4)   |   input;
// Add ones and return answer
}


/*************************************
*************************************
*
*   Function name : LCD_Init
*
*   Returns :       None
*
*   Parameters :    None
*
*     Purpose :             Initialize
LCD_displayData buffer.
*                        Set up the LCD
(timing, contrast, etc.)
*
*************************************
*************************************/
void LCD_Init (void)
{
    LCD_AllSegments(FALSE);
// Clear segment buffer.


LCD_CONTRAST_LEVEL(LCD_INITIAL_CONTRAST
);    //Set the LCD contrast level
```

```
    //    Select    asynchronous    clock
source, enable all COM pins and enable
all
    // segment pins.
    LCDCRB = (1<<LCDCS) | (3<<LCDMUX0)
| (7<<LCDPM0);

    //  Set  LCD  prescaler  to  give  a
framerate of 32,0 Hz
    LCDFRR = (0<<LCDPS0) | (7<<LCDCD0);

    LCDCRA = (1<<LCDEN) | (1<<LCDAB);
//  Enable  LCD  and  set  low  power
waveform

    //Enable   LCD   start   of   frame
interrupt
    LCDCRA |= (1<<LCDIE);

    gLCD_Update_Required = FALSE;


}


/*************************************
*************************************
*
*    Function name : LCD_WriteDigit(char
c, char digit)
*
*   Returns :       None
*
*   Parameters :    Inputs
*                    c: The symbol to be
displayed in a LCD digit
*                        digit: In which
digit   (0-5)   the   symbol   should   be
displayed
*                      Note: Digit 0 is
the first used digit on the LCD,
*                   i.e LCD digit 2
*
*    Purpose :        Stores LCD control
data in the LCD_displayData buffer.
*                                    (The
LCD_displayData   is   latched   in   the
LCD_SOF interrupt.)
*
*************************************
*************************************/
void LCD_WriteDigit(char c, char digit)
{

    unsigned   int   seg   =   0x0000;
// Holds the segment pattern
    char mask, nibble;
    char *ptr;
    char i;


    if        (digit        >        5)
// Skip if digit is illegal
        return;

    //Lookup character table for segmet
data
    if ((c >= '*') && (c <= 'z'))
    {
        // c is a letter
        if       (c       >=       'a')
// Convert to upper case
            c          &=          ~0x20;
// if necessarry

        c -= '*';

            //mt          seg          =
LCD_character_table[c];
            seg   =   (unsigned   int)
pgm_read_word(&LCD_character_table[(uin
t8_t)c]);
      }

    //  Adjust  mask  according  to  LCD
segment mapping
    if (digit & 0x01)
        mask = 0x0F;                    //
Digit 1, 3, 5
    else
        mask = 0xF0;                    //
Digit 0, 2, 4

    ptr = LCD_Data + (digit >> 1);   //
digit = {0,0,1,1,2,2}

    for (i = 0; i < 4; i++)
    {
        nibble = seg & 0x000F;
        seg >>= 4;
        if (digit & 0x01)
            nibble <<= 4;
        *ptr = (*ptr & mask) | nibble;
        ptr += 5;
    }
}



/*************************************
*************************************
*
*                Function     name     :
LCD_AllSegments(unsigned char input)
*
*   Returns :       None
*
*       Parameters   :          show   -
[TRUE;FALSE]
*
*    Purpose :        shows or hide all
all LCD segments on the LCD
*
*************************************
*************************************/
void LCD_AllSegments(char show)
```

```
{
    unsigned char i;

    if (show)
        show = 0xFF;

    // Set/clear all bits in all LCD
registers
    for (i=0; i < LCD_REGISTER_COUNT;
i++)
        *(LCD_Data + i) = show;
}


/*************************************
*************************************
*
*   LCD Interrupt Routine
*
*   Returns :       None
*
*   Parameters :    None
*
*    Purpose: Latch the LCD_displayData
and Set LCD_status.updateComplete
*
*************************************
*************************************/

SIGNAL(SIG_LCD)
{
    static     char     LCD_timer     =
LCD_TIMER_SEED;
    char c;
    char c_flash;
    char flash;

    char EOL;
    unsigned char i;

///!!!    static char timeout_count;
///!!!                    static     char
auto_joystick_count;

       c_flash=0; // mt

/***************  Button  timeout  for
the button.c, START ***************/
/*!!!    if(!gButtonTimeout)
    {
        timeout_count++;

        if(timeout_count > 3)
        {
            gButtonTimeout = TRUE;
            timeout_count = 0;
        }
    }
*/
/***************  Button  timeout  for
the button.c, END *****************/

/***************  Auto  press  joystick
for the main.c, START *************/

/*!!!    if(gAutoPressJoystick == AUTO)
    {
        auto_joystick_count++;

        if(auto_joystick_count > 16)
        {
            gAutoPressJoystick = TRUE;
            auto_joystick_count = 15;
        }
    }
    else
        auto_joystick_count = 0;
*/

/***************  Auto  press  joystick
for the main.c, END ***************/

    LCD_timer--;                        //
Decreased every LCD frame

    if (gScrollMode)
    {
        // If we are in scroll mode,
and the timer has expired,
        // we will update the LCD
        if (LCD_timer == 0)
        {
            if  (gLCD_Start_Scroll_Timer
== 0)
            {
                gLCD_Update_Required   =
TRUE;
            }
            else

gLCD_Start_Scroll_Timer--;
        }
    }
    else
    {   // if not scrolling,
        //  disble  LCD  start  of  frame
interrupt
//        cbi(LCDCRA, LCDIE);   //DEBUG
        gScroll = 0;
    }


    EOL = FALSE;
    if (gLCD_Update_Required == TRUE)
    {
        //   Duty   cycle   of   flashing
characters
        if        (gFlashTimer        <
(LCD_FLASH_SEED >> 1))
            flash = 0;
        else
            flash = 1;

        //   Repeat   for   the   six   LCD
characters
```

```
        for (i = 0; i < 6; i++)
        {
            if  ((gScroll+i)  >=  0  &&
(!EOL))
            {
                // We have some visible
characters
                c   =   gTextBuffer[i   +
gScroll];
                c_flash = c & 0x80 ? 1
: 0;
                c = c & 0x7F;

                if (c == '\0')
                    EOL = i+1;        //
End of character data
            }
            else
                c = ' ';

            // Check if this character
is flashing

            if (c_flash && flash)
                LCD_WriteDigit(' ', i);
            else
                LCD_WriteDigit(c, i);
        }

        // Copy the segment buffer to
the real segments
        for     (i     =     0;     i     <
LCD_REGISTER_COUNT; i++)
            *(pLCDREG     +     i)     =
*(LCD_Data+i);

        // Handle colon
        if (gColon)
            *(pLCDREG + 8) = 0x01;
        else
            *(pLCDREG + 8) = 0x00;

        // If the text scrolled off the
display,
        // we have to start over again.
        if (EOL == 1)
            gScroll = -6;
        else
            gScroll++;

        // No need to update anymore
        gLCD_Update_Required = FALSE;
    }


    // LCD_timer is used when scrolling
text
    if (LCD_timer == 0)
    {
/*        if ((gScroll <= 0) || EOL)
            LCD_timer                   =
LCD_TIMER_SEED/2;
        else*/
            LCD_timer = LCD_TIMER_SEED;
    }

    //   gFlashTimer   is   used   when
flashing characters
    if (gFlashTimer == LCD_FLASH_SEED)
        gFlashTimer= 0;
    else
        gFlashTimer++;

}


/*************************************
*************************************
*
*
*   Function name:  RTC_init
*
*   returns:        none
*
*   parameters:     none
*
*       Purpose:                    Start
Timer/Counter2      in      asynchronous
operation using a
*                   32.768kHz crystal.
*
*************************************
*************************************
*/
void RTC_init(void)
{
    Delay(1000);                  // wait
for  1  sec  to  let  the  Xtal  stabilize
after a power-on,

    cli();                              //
disable global interrupt

    cbi(TIMSK2,  TOIE2);               //
disable OCIE2A and TOIE2

    ASSR = (1<<AS2);           // select
asynchronous operation of Timer2

    TCNT2 = 0;                   // clear
TCNT2A
    TCCR2A  |=  (1<<CS22)  |  (1<<CS20);
// select precaler: 32.768 kHz / 128 =
1 sec between each overflow

    while((ASSR   &   0x01)   |   (ASSR   &
0x04));          // wait for TCN2UB and
TCR2UB to be cleared

    TIFR2 = 0xFF;                // clear
interrupt-flags
    sbi(TIMSK2, TOIE2);         // enable
Timer2 overflow interrupt

    sei();                       // enable
global interrupt
```

```
    // initial time and date setting
    gSECOND  = 0;
    gMINUTE  = 0;
    gHOUR    = 12;
    // mt release timestamp
    gDAY     = 16;
    gMONTH   = 10;
    gYEAR    = 3;
}



/*************************************
*************************************
*
*
*     Timer/Counter2  Overflow  Interrupt
Routine
*
*     Purpose:  Increment  the  real-time
clock
*              The interrupt occurs once
a   second   (running   from   the   32kHz
crystal)
*                  This  routine  remains
active during the Power Save sleep mode
*************************************
*************************************
*/
SIGNAL(SIG_OVERFLOW2)
{
    static char LeapMonth;

    gSECOND++;                          //
increment second

    gPowerSaveTimer++;                  //
increment "time until sleep" timer
    gWakeUpTimer++;                     //
increment "time until wake up" timer


    if (gSECOND == 60)
    {
        gSECOND = 0;
        gMINUTE++;

        if (gMINUTE > 59)
        {
            gMINUTE = 0;
            gHOUR++;

            if (gHOUR > 23)
            {

                gHOUR = 0;
                gDAY++;

                // Check for leap year
if month == February
                if (gMONTH == 2)
                    if     (!(gYEAR    &
0x0003))               // if (gYEAR%4 ==
0)
                        if    (gYEAR%100
== 0)
                            if
(gYEAR%400 == 0)

LeapMonth = 1;
                            else

LeapMonth = 0;
                        else
                            LeapMonth  =
1;
                    else
                        LeapMonth = 0;
                else
                    LeapMonth = 0;

                //  Now,  we  can  check
for month length
                if        (gDAY        >
(MonthLength[gMONTH] + LeapMonth))
                {
                    gDAY = 1;
                    gMONTH++;

                    if (gMONTH > 12)
                    {
                        gMONTH = 1;
                        gYEAR++;
                    }
                }
            }
        }
    }
}



/*************************************
*************************************
*
*   Function name : Timer0_Init
*
*   Returns :       None
*
*   Parameters :    None
*
*     Purpose  :             Initialize
Timer/Counter 0
*
*************************************
*************************************/
void Timer0_Init(void)
{
    //mt char i;
       uint8_t i;

    //  Initialize  array  of  callback
functions
    for   (i=0;   i<TIMER0_NUM_CALLBACKS;
i++)
```

```
        CallbackFunc[i] = NULL;

    // Initialize countdown timers
    for                              (i=0;
i<TIMER0_NUM_COUNTDOWNTIMERS; i++)
        CountDownTimers[i] = 255;


       // Initialize Timer0.
       // Used to give the correct time-
delays in the song

    // Enable timer0 compare interrupt
       TIMSK0 = (1<<OCIE0A);

    // Sets the compare value for 10ms
delay
#ifdef OSC_ALT
       OCR0A = 35;   // t0 is at 3600kHz
#else
       OCR0A = 38; // t0 is ~3900kHz
#endif

       //  Set  Clear  on  Timer  Compare
(CTC) mode, CLK/256 prescaler
       //     TCCR0A                    =
(1<<WGM01)|(0<<WGM00)|(4<<CS00);      //
1Mhz Clk

       //  Set  Clear  on  Timer  Compare
(CTC) mode, CLK/1024 prescaler
       TCCR0A                           =
(1<<WGM01)|(0<<WGM00)|(5<<CS00);      //
4Mhz Clk
}



/*************************************
*************************************
*
*                Function     name     :
TIMER0_COMP_interrupt
*
*   Returns :       None
*
*   Parameters :    None
*
*     Purpose  :           Check  if  any
functions are to be called
*
*************************************
*************************************/
// mtA
// #pragma vector = TIMER0_COMP_vect
//            __interrupt           void
TIMER0_COMP_interrupt(void)
SIGNAL(SIG_OUTPUT_COMPARE0)
// mtE
{
    // mt char i;
       uint8_t i;

    for   (i=0;   i<TIMER0_NUM_CALLBACKS;
i++)
        if (CallbackFunc[i] != NULL)
            CallbackFunc[i]();

    // Count down timers
    for                              (i=0;
i<TIMER0_NUM_COUNTDOWNTIMERS; i++)
        if  (CountDownTimers[i]  !=  255
&& CountDownTimers[i] != 0)
            CountDownTimers[i]--;

}


/*************************************
*************************************
*
*                Function     name     :
Timer0_RegisterCallbackFunction
*
*   Returns :       None
*
*   Parameters :    pFunc
*
*   Purpose :        Set up functions to
be called from the
*
TIMER0_COMP_interrupt
*
*************************************
*************************************/
BOOL
Timer0_RegisterCallbackFunction(TIMER_C
ALLBACK_FUNC pFunc)
{
    // mt char i;
       uint8_t i;

    for   (i=0;   i<TIMER0_NUM_CALLBACKS;
i++)
    {
        if (CallbackFunc[i] == pFunc)
            return TRUE;
    }

    for   (i=0;   i<TIMER0_NUM_CALLBACKS;
i++)
    {
        if (CallbackFunc[i] == NULL)
        {
            CallbackFunc[i] = pFunc;
            return TRUE;
        }
    }

    return FALSE;
}


/*************************************
*************************************
*
```

```
*                Function     name     :
Timer0_RemoveCallbackFunction
*
*   Returns :       None
*
*   Parameters :    pFunc
*
*    Purpose :          Remove functions
from the list which is called int the
*
TIMER0_COMP_interrupt
*
*************************************
*************************************/
BOOL
Timer0_RemoveCallbackFunction(TIMER_CAL
LBACK_FUNC pFunc)
{
    // mt char i;
       uint8_t i;

    for   (i=0;   i<TIMER0_NUM_CALLBACKS;
i++)
    {
        if (CallbackFunc[i] == pFunc)
        {
            CallbackFunc[i] = NULL;
            return TRUE;
        }
    }

    return FALSE;
}


char Timer0_AllocateCountdownTimer()
{
    // mt char i;
       uint8_t i;

    for                              (i=0;
i<TIMER0_NUM_COUNTDOWNTIMERS; i++)
        if (CountDownTimers[i] == 255)
        {
            CountDownTimers[i] = 0;
            return i+1;
        }

    return 0;
}

void       Timer0_SetCountdownTimer(char
timer, char value)
{
    cli(); // mt __disable_interrupt();
    CountDownTimers[timer-1] = value;
    sei(); // mt __enable_interrupt();
}

char       Timer0_GetCountdownTimer(char
timer)
{
    char t;
    cli(); // mt __disable_interrupt();
    t = CountDownTimers[timer-1];
    sei(); // mt __enable_interrupt();

    return t;
}

void   Timer0_ReleaseCountdownTimer(char
timer)
{
    cli(); // mt __disable_interrupt();
    CountDownTimers[timer-1] = 255;
    sei(); // mt __enable_interrupt();
}
```

```
 "main.h"


//************************************
************************************
//
//  File.........: main.h
//
//  Author(s)...: Achyut Shrestha
//
//  Target(s)...: ATmega169
//
//   Compiler....: AVR-GCC 3.3.1; avr-
libc 1.2
//
//    Description.: AVR Butterfly main
routines
//
//  Revisions...: 1.0
//
//      YYYYMMDD   -   VER.   -   COMMENT
- SIGN.
//
//  20080503    main  routine    -  Achyut
Shrestha
//
//************************************
************************************

#define BOOL    char

#define FALSE   0
```

```
#define TRUE    (!FALSE)
#define NULL    0

#define AUTO    3

#define _BV(bit)                (1   <<
(bit))
#define sbi(x,bit)              (x   |=
_BV(bit))   //   set   bit      (   GNU
compatability )
#define cbi(x,bit)              (x   &=
~_BV(bit))   //   clear   bit    (   GNU
compatability )


//Button definitions
//
#define KEY_NULL    0
#define KEY_ENTER   1
#define KEY_NEXT    2
#define KEY_PREV    3
#define KEY_PLUS    4
#define KEY_MINUS   5

void Initialization(void);

void OSCCAL_calibration(void);

void Delay(unsigned int millisec);

#define TEMPERATURE_SENSOR  0
#define VOLTAGE_SENSOR      1
#define LIGHT_SENSOR        2
#define CELCIUS             3
#define FARENHEIT           4
#define DIR_SENSOR         4

// this is effeted by the light and
temp a little
// 1.07 is a compromise so analog
reading can vary by +/- 50mV
// this is okay as we are only using it
to moonitor the battery voltage
#define VREF        2.900

#define PWR_MAX_V          0xBC     ///<
3.2V
#define PWR_GOOD_V         0xB0     ///<
3.0V
#define PWR_LOW_V          0x9E     ///<
2.7V average voltage
#define PWR_MIN_V          0x8D///< 2.4V
#define PWR_ABS_MIN_V      0x75///< 2.0V
absolute   minimum   allowed   battery
voltage

void ADC_init(char );
int ADC_read(void);
void ADC2Temp(int ADCResult);
void ADC2Volt(int ADCResult);
void Power_check(void);
char Temp_check(void);
```

```
#ifdef M162
#define                         PINA_MASK
((1<<PINA0)|(1<<PINA1)|(1<<PINA2)|(1<<P
INA3)|(1<<PINA4))
#else
#define                         PINB_MASK
((1<<PINB4)|(1<<PINB6)|(1<<PINB7))
#define                         PINE_MASK
((1<<PINE2)|(1<<PINE3)|(1<<PINE4))

#endif

#ifdef M162
#define BUTTON_A    0   // NORTH
#define BUTTON_B    1   // EAST
#define BUTTON_C    2   // WEST
#define BUTTON_D    3   // SOUTH
#define BUTTON_O    4   // PUSH
#else
#define BUTTON_A    6   // UP
#define BUTTON_B    7   // DOWN
#define BUTTON_C    2   // LEFT
#define BUTTON_D    3   // RIGHT
#define BUTTON_O    4   // PUSH
#endif

void PinChangeInterrupt(void);
void Button_Init(void);
char getkey(void);
char ButtonBouncing(void);


// extern char gPowerSaveTimer;
extern volatile char gPowerSaveTimer;

extern  uint16_t  PowerSaveTimeout;    //
Initial value, sleep after 30 sec

// mt   unsigned char gButtonTimeout =
FALSE;
volatile unsigned char gButtonTimeout =
FALSE;


volatile char KEY = NULL;
volatile char KEY_VALID = FALSE;



#include <avr/pgmspace.h>
//Functions
//   mt   void   LCD_puts_f(char   __flash
*pFlashStr, char scrollmode);
void  LCD_puts_f(const  char  *pFlashStr,
char scrollmode);
void    LCD_puts(char    *pStr,    char
scrollmode);
void   LCD_UpdateRequired(char   update,
char scrollmode);
//void    LCD_putc(char    digit,    char
character);
```

```
void   LCD_putc(uint8_t   digit,   char
character);
void LCD_Clear(void);
void LCD_Colon(char show);
void LCD_FlashReset(void);
char SetContrast(char input);



// Function declarations
char CHAR2BCD2(char input);
unsigned int CHAR2BCD3(char input);



/************************************
*********************************/
// Definitions
/************************************
*********************************/
#define LCD_INITIAL_CONTRAST    0x0F
#define LCD_TIMER_SEED                3
#define LCD_FLASH_SEED          10
#define LCD_REGISTER_COUNT      20
#define TEXTBUFFER_SIZE         25

#define SCROLLMODE_ONCE         0x01
#define SCROLLMODE_LOOP         0x02
#define SCROLLMODE_WAVE         0x03

/************************************
*********************************/
//MACROS
/************************************
*********************************/
//active = [TRUE;FALSE]
#define            LCD_SET_COLON(active)
LCD_Data[8] = active

// DEVICE SPECIFIC!!! (ATmega169)
#define    pLCDREG    ((unsigned    char
*)(0xEC))

// DEVICE SPECIFIC!!! (ATmega169) First
LCD segment register
#define        LCD_CONTRAST_LEVEL(level)
LCDCCR=(0x0F & level)



/************************************
*********************************/
// Global variables
/************************************
*********************************/
extern           volatile           char
gLCD_Update_Required;
extern                              char
LCD_Data[LCD_REGISTER_COUNT];
extern                              char
gTextBuffer[TEXTBUFFER_SIZE];
extern volatile char gScrollMode;
extern char gFlashTimer;
extern char gColon;
extern volatile signed char gScroll;


/************************************
*********************************/
// Global functions
/************************************
*********************************/
void LCD_Init (void);
void  LCD_WriteDigit(char  input,  char
digit);
void LCD_AllSegments(char show);

extern uint8_t gSECOND;
extern uint8_t gMINUTE;
extern uint8_t gHOUR;
extern uint8_t gDAY;
extern uint8_t gMONTH;
//mtE
extern unsigned int gYEAR;


//  Function declarations
void                      RTC_init(void);
//initialize  the  Timer  Counter  2  in
asynchron operation

#define CLOCK_24    1
#define CLOCK_12    0

// Typedef for Timer callback function
typedef    void    (*TIMER_CALLBACK_FUNC)
(void);

#define TIMER0_NUM_CALLBACKS        4
#define TIMER0_NUM_COUNTDOWNTIMERS  4

void Timer0_Init(void);

BOOL
Timer0_RegisterCallbackFunction(TIMER_C
ALLBACK_FUNC pFunc);
BOOL
Timer0_RemoveCallbackFunction(TIMER_CAL
LBACK_FUNC pFunc);


//mt
//                                    char
Timer0_AllocateCountdownTimer();
char
Timer0_AllocateCountdownTimer(void);
char       Timer0_GetCountdownTimer(char
timer);
void       Timer0_SetCountdownTimer(char
timer, char value);
void   Timer0_ReleaseCountdownTimer(char
timer);


“makefile”
```

```
# WinAVR Sample makefile written by Eric B. Weddington, Jörg Wunsch, et al.
# Released to the Public Domain
# Please read the make user manual!
#
# Additional material for this makefile was submitted by:
#  Tim Henigan
#  Peter Fleury
#  Reiner Patommel
#  Sander Pool
#  Frederik Rouleau
#  Markus Pfaff
#
# On command line:
#
# make all = Make software.
#
# make clean = Clean out built project files.
#
# make coff = Convert ELF to AVR COFF (for use with AVR Studio 3.x or VMLAB).
#
# make extcoff = Convert ELF to AVR Extended COFF (for use with AVR Studio
#                 4.07 or greater).
#
# make program = Download the hex file to the device, using avrdude.  Please
#                 customize the avrdude settings below first!
#
# make filename.s = Just compile filename.c into the assembler code only
#
# To rebuild project do "make clean" then "make all".
#


# MCU name
MCU = atmega169

# Output format. (can be srec, ihex, binary)
FORMAT = ihex

# Target file name (without extension).
TARGET = main

# Optimization level, can be [0, 1, 2, 3, s]. 0 turns off optimization.
# (Note: 3 is not always the best optimization level. See avr-libc FAQ.)
#OPT = s
OPT=s
#OPT = 0

# List C source files here. (C dependencies are automatically generated.)
SRC = $(TARGET).c

# If there is more than one source file, append them above, or modify and
# uncomment the following:
#   the file test.c can be omitted (remove #inc test.h and test() in main.c)



# You can also wrap lines by appending a backslash to the end of the line:
#SRC += baz.c \
#xyzzy.c


# List Assembler source files here.
# Make them always end in a capital .S.  Files ending in a lowercase .s
# will not be considered source files but generated files (assembler
# output from the compiler), and will be deleted upon "make clean"!
# Even though the DOS/Win* filesystem matches both .s and .S the same,
# it will preserve the spelling of the filenames, and gcc itself does
# care about how the name is spelled on its command-line.
ASRC =


# List any extra directories to look for include files here.
#      Each directory must be seperated by a space.
EXTRAINCDIRS =


# Optional compiler flags.
#    -g:              generate debugging information (for GDB, or for COFF conversion)
#  -O*:       optimization level
#   -f...:       tuning, see gcc manual and avr-libc documentation
#  -Wall...:  warning level
#   -Wa,...:    tell GCC to pass this to the assembler.
#    -ahlms:  create assembler listing
CFLAGS = -g -O$(OPT) \
-funsigned-char -funsigned-bitfields -fpack-struct -fshort-enums \
-Wall -Wstrict-prototypes \
-Wa,-adhlns=$(<:.c=.lst) \
$(patsubst %,-I%,$(EXTRAINCDIRS))


# Set a "language standard" compiler flag.
```

```
#   Unremark just one line below to set
the language standard to use.
#    gnu99 = C99 + GNU extensions. See
GCC manual for more information.
#CFLAGS += -std=c89
#CFLAGS += -std=gnu89
#CFLAGS += -std=c99
CFLAGS += -std=gnu99



# Optional assembler flags.
#  -Wa,...:    tell GCC to pass this to
the assembler.
#  -ahlms:    create listing
#  -gstabs:    have the assembler create
line number information; note that
#                for use in COFF files,
additional information about filenames
#               and function names needs
to be present in the assembler source
#                files -- see avr-libc
docs [FIXME: not yet described there]
ASFLAGS   =   -Wa,-adhlns=$(<:.S=.lst),-
gstabs



# Optional linker flags.
#  -Wl,...:    tell GCC to pass this to
linker.
#  -Map:      create map file
#   --cref:      add cross reference to
map file
LDFLAGS = -Wl,-Map=$(TARGET).map,--cref



# Additional libraries

# Minimalistic printf version
#LDFLAGS     +=     -Wl,-u,vfprintf     -
lprintf_min

#   Floating   point   printf   version
(requires -lm below)
#LDFLAGS     +=     -Wl,-u,vfprintf     -
lprintf_flt

# -lm = math library
LDFLAGS += -lm



#  Programming  support  using  avrdude.
Settings and variables.

#  Programming  hardware:  alf  avr910
avrisp bascom bsd
#  dt006  pavr  picoweb  pony-stk200  sp12
stk200 stk500
#
# Type: avrdude -c ?
# to get a full listing.
#
AVRDUDE_PROGRAMMER = stk500


AVRDUDE_PORT = com1    #      programmer
connected to serial device
#AVRDUDE_PORT = lpt1       #   programmer
connected to parallel port

AVRDUDE_WRITE_FLASH          =          -U
flash:w:$(TARGET).hex
#AVRDUDE_WRITE_EEPROM         =         -U
eeprom:w:$(TARGET).eep

AVRDUDE_FLAGS    =    -p    $(MCU)    -P
$(AVRDUDE_PORT)                         -c
$(AVRDUDE_PROGRAMMER)

# Uncomment the following if you want
avrdude's erase cycle counter.
# Note that this counter needs to be
initialized first using -Yn,
# see avrdude manual.
#AVRDUDE_ERASE += -y

#  Uncomment  the  following  if  you  do
/not/ wish a verification to be
#   performed   after   programming   the
device.
#AVRDUDE_FLAGS += -V

# Increase verbosity level.  Please use
this when submitting bug
#    reports    about    avrdude.    See
<http://savannah.nongnu.org/projects/av
rdude>
# to submit bug reports.
#AVRDUDE_FLAGS += -v -v



#  ------------------------------------
------------------------------------

# Define directories, if needed.
DIRAVR = c:/winavr
DIRAVRBIN = $(DIRAVR)/bin
DIRAVRUTILS = $(DIRAVR)/utils/bin
DIRINC = .
DIRLIB = $(DIRAVR)/avr/lib


# Define programs and commands.
SHELL = sh

CC = avr-gcc

OBJCOPY = avr-objcopy
OBJDUMP = avr-objdump
SIZE = avr-size
```

```
# Programming support using avrdude.
AVRDUDE = avrdude


REMOVE = rm -f
COPY = cp

HEXSIZE = $(SIZE) --target=$(FORMAT)
$(TARGET).hex
#mt - use hexadezimal output-fromat //
org:       ELFSIZE    =    $(SIZE)    -A
$(TARGET).elf
ELFSIZE = $(SIZE) -x -A $(TARGET).elf


# Define Messages
# English
MSG_ERRORS_NONE = Errors: none
MSG_BEGIN = -------- begin --------
MSG_END = --------  end  --------
MSG_SIZE_BEFORE = Size before:
MSG_SIZE_AFTER = Size after:
MSG_COFF = Converting to AVR COFF:
MSG_EXTENDED_COFF = Converting to AVR
Extended COFF:
MSG_FLASH = Creating load file for
Flash:
MSG_EEPROM = Creating load file for
EEPROM:
MSG_EXTENDED_LISTING     =     Creating
Extended Listing:
MSG_SYMBOL_TABLE   =   Creating   Symbol
Table:
MSG_LINKING = Linking:
MSG_COMPILING = Compiling:
MSG_ASSEMBLING = Assembling:
MSG_CLEANING = Cleaning project:



# Define all object files.
OBJ = $(SRC:.c=.o) $(ASRC:.S=.o)

# Define all listing files.
LST = $(ASRC:.S=.lst) $(SRC:.c=.lst)

#  Combine  all  necessary  flags  and
optional flags.
# Add target processor to flags.
ALL_CFLAGS = -mmcu=$(MCU) -I. $(CFLAGS)
ALL_ASFLAGS   =   -mmcu=$(MCU)   -I.   -x
assembler-with-cpp $(ASFLAGS)



# Default target.
all:   begin   gccversion   sizebefore
$(TARGET).elf             $(TARGET).hex
$(TARGET).eep \
      $(TARGET).lss       $(TARGET).sym
sizeafter finished end


# Eye candy.
# AVR Studio 3.x does not check make's
exit code but relies on
#  the  following  magic  strings  to  be
generated by the compile job.
begin:
      @echo
      @echo $(MSG_BEGIN)

finished:
      @echo $(MSG_ERRORS_NONE)

end:
      @echo $(MSG_END)
      @echo


# Display size of file.
sizebefore:
      @if  [  -f  $(TARGET).elf  ];  then
echo;      echo      $(MSG_SIZE_BEFORE);
$(ELFSIZE); echo; fi

sizeafter:
      @if  [  -f  $(TARGET).elf  ];  then
echo;      echo      $(MSG_SIZE_AFTER);
$(ELFSIZE); echo; fi



# Display compiler version information.
gccversion :
      @$(CC) --version



#  Convert  ELF  to  COFF  for  use  in
debugging / simulating in
# AVR Studio or VMLAB.
COFFCONVERT=$(OBJCOPY) --debugging \
      --change-section-address   .data-
0x800000 \
      --change-section-address    .bss-
0x800000 \
      --change-section-address .noinit-
0x800000 \
      --change-section-address .eeprom-
0x810000


coff: $(TARGET).elf
      @echo
      @echo $(MSG_COFF) $(TARGET).cof
      $(COFFCONVERT)  -O  coff-avr  $<
$(TARGET).cof


extcoff: $(TARGET).elf
```

```
	@echo
	@echo		$(MSG_EXTENDED_COFF)
$(TARGET).cof
	$(COFFCONVERT) -O coff-ext-avr $<
$(TARGET).cof



# Program the device.
program: $(TARGET).hex $(TARGET).eep
	$(AVRDUDE)	$(AVRDUDE_FLAGS)
$(AVRDUDE_WRITE_FLASH)
$(AVRDUDE_WRITE_EEPROM)



# Create final output files (.hex,
.eep) from ELF output file.
%.hex: %.elf
	@echo
	@echo $(MSG_FLASH) $@
	$(OBJCOPY) -O $(FORMAT) -R
.eeprom $< $@

%.eep: %.elf
	@echo
	@echo $(MSG_EEPROM) $@
	-$(OBJCOPY) -j .eeprom --set-
section-flags=.eeprom="alloc,load" \
	--change-section-lma .eeprom=0 -O
$(FORMAT) $< $@

# Create extended listing file from ELF
output file.
%.lss: %.elf
	@echo
	@echo $(MSG_EXTENDED_LISTING) $@
	$(OBJDUMP) -h -S $< > $@

# Create a symbol table from ELF output
file.
%.sym: %.elf
	@echo
	@echo $(MSG_SYMBOL_TABLE) $@
	avr-nm -n $< > $@



# Link: create ELF output file from
object files.
.SECONDARY : $(TARGET).elf
.PRECIOUS : $(OBJ)
%.elf: $(OBJ)
	@echo
	@echo $(MSG_LINKING) $@
	$(CC) $(ALL_CFLAGS) $(OBJ) --
output $@ $(LDFLAGS)


# Compile: create object files from C
source files.
%.o : %.c
	@echo
	@echo $(MSG_COMPILING) $<
	$(CC) -c $(ALL_CFLAGS) $< -o $@


# Compile: create assembler files from
C source files.
%.s : %.c
	$(CC) -S $(ALL_CFLAGS) $< -o $@


# Assemble: create object files from
assembler source files.
%.o : %.S
	@echo
	@echo $(MSG_ASSEMBLING) $<
	$(CC) -c $(ALL_ASFLAGS) $< -o $@



# Target: clean project.
clean: begin clean_list finished end

clean_list :
	@echo
	@echo $(MSG_CLEANING)
	$(REMOVE) $(TARGET).hex
	$(REMOVE) $(TARGET).eep
	$(REMOVE) $(TARGET).obj
	$(REMOVE) $(TARGET).cof
	$(REMOVE) $(TARGET).elf
	$(REMOVE) $(TARGET).map
	$(REMOVE) $(TARGET).obj
	$(REMOVE) $(TARGET).a90
	$(REMOVE) $(TARGET).sym
	$(REMOVE) $(TARGET).lnk
	$(REMOVE) $(TARGET).lss
	$(REMOVE) $(OBJ)
	$(REMOVE) $(LST)
	$(REMOVE) $(SRC:.c=.s)
	$(REMOVE) $(SRC:.c=.d)

# Automatically generate C source code
dependencies.
# (Code originally taken from the GNU
make user manual and modified
# (See README.txt Credits).)
#
# Note that this will work with sh
(bash) and sed that is shipped with
WinAVR
# (see the SHELL variable defined
above).
# This may not work with other shells
or other seds.
#
%.d: %.c
	set -e; $(CC) -MM $(ALL_CFLAGS)
$< \
```

```
	| sed 's,\(.*\)\.o[ :]*,\1.o \1.d
: ,g' > $@; \
	[ -s $@ ] || rm -f $@


# Remove the '-' if you want to see the
dependency files generated.
-include $(SRC:.c=.d)
```

```
# Listing of phony targets.
.PHONY : all begin finish end
sizebefore sizeafter gccversion coff
extcoff \
	clean clean_list program
```

## 10.10. Appendix X – Regulator Test Results

| | | Voltages on the MAX1524 | | | | | |
|---|---|---|---|---|---|---|---|
| $V_{in}$ | $V_{out}$ | $V_{GND}$ | $V_{FB}$ | $V_{SET}$ | $V_{SHDN}$ | $V_{EXT}$ | $V_{CC}$ |
| 0.25 | 0.187 | 0 | 0 | 0 | 0.25 | 0 | 0.25 |
| 0.5 | 0.437 | 0 | 0 | 0 | 0.5 | 0 | 0.5 |
| 0.75 | 0.625 | 0 | 0 | 0 | 0.75 | 0 | 0.75 |
| 1 | 1.12 | 0 | 0.437 | 0 | 1 | 1 | 1 |
| 1.25 | 2.62 | 0 | 1.37 | 0 | 1.12 | 1.12 | 1.12 |
| 1.5 | 4.37 | 0 | 2.5 | 0 | 1.25 | 1.25 | 1.25 |
| 1.75 | 7.93 | 0 | 3.56 | 0 | 1.31 | 1.37 | 1.31 |
| 2 | 7.93 | 0 | 1.68 | 0 | 1.37 | 1.37 | 1.37 |
| 2.25 | 7.93 | 0 | 5 | 0 | 1.37 | 1.37 | 1.37 |
| 2.5 | 7.93 | 0 | 7.93 | 0 | 1.43 | 1.43 | 1.43 |
| 2.75 | 7.93 | 0 | 7 | 0 | 1.43 | 1.43 | 1.43 |
| 3 | 5 | 0 | 2.18 | 0 | 1.06 | 1.06 | 1.06 |

| R | $I_{out}$ |
|---|---|
| 988 | 4.82 |
| 5047 | 4.93 |
| 10886 | 1.25 |
| 50360 | 1.25 |
| 98800 | 1.25 |
| 520000 | 1.25 |
| 977000 | 1.25 |

| Resistor Voltages | | | Capacitor Voltages | | | | Inductor Voltage |
|---|---|---|---|---|---|---|---|
| $V_{R1}$ | $V_{R2}$ | $V_{R3}$ | $V_{C1}$ | $V_{C2}$ | $V_{C3}$ | $V_{CFF}$ | $V_{L1}$ |
| 0.187 | 0 | 0.25 | 0.25 | 0.187 | 0.25 | 0.187 | 0.25 |
| 0.437 | 0 | 0.5 | 0.5 | 0.437 | 0.5 | 0.437 | 0.5 |
| 0.625 | 0.187 | 0.75 | 0.75 | 0.625 | 0.75 | 0.625 | 0.75 |
| 1.12 | 0.437 | 1 | 1 | 1.12 | 1 | 1.12 | 1 |
| 2.62 | 1.37 | 1.25 | 1.25 | 2.62 | 1 | 0.875 | 1.25 |
| 4.25 | 2.5 | 1.5 | 1.5 | 4.31 | 1.18 | 1.93 | 1.56 |
| 6.6 | 4 | 1.87 | 1.75 | 7.93 | 1.31 | 3.25 | 1.75 |
| 7.93 | 6.81 | 2.08 | 2 | 7.93 | 1.37 | 4.75 | 2 |
| 7.93 | 5 | 2.12 | 2.18 | 7.93 | 1.37 | 7.93 | 2 |
| 7.93 | 7.93 | 2.5 | 2.43 | 7.93 | 1.43 | 7.93 | 2.43 |
| 7.93 | 7.12 | 2.62 | 2.5 | 7.93 | 1.5 | 7.93 | 2.56 |
| 3.37 | 2.12 | 3 | 3 | 4.93 | 1.06 | 4.93 | 2.87 |

| MOSFET Voltages | | | Diode Voltages | |
|---|---|---|---|---|
| $V_{Q1S}$ | $V_{Q1G}$ | $V_{Q1D}$ | $V_{D1in}$ | $V_{D1out}$ |
| 0 | 0 | 0.25 | 0.25 | 0.187 |
| 0 | 0 | 0.5 | 0.5 | 0.437 |
| 0 | 0 | 0.75 | 0.75 | 0.625 |
| 0 | 1 | 1.25 | 1.25 | 1.12 |
| 0 | 1.12 | 2.87 | 2.68 | 2.62 |
| 0 | 1.25 | 4.37 | 4.5 | 4.37 |
| 0 | 1.37 | 5.43 | 5.31 | 4.81 |
| 0 | 1.37 | 5.93 | 5.87 | 5.18 |
| 0 | 1.37 | 7.93 | 7.93 | 7.25 |
| 0 | 1.43 | 7.93 | 7.93 | 7.93 |
| 0 | 1.43 | 7 | 7.93 | 7.93 |
| 0 | 1.06 | 3.81 | 5.06 | 4.75 |

| in mA | | Current on the MAX1524 | | | | | |
|---|---|---|---|---|---|---|---|
| $I_{in}$ | $I_{out}$ | $I_{GND}$ | $I_{FB}$ | $I_{SET}$ | $I_{SHDN}$ | $I_{EXT}$ | $I_{CC}$ |
| 312 | 187 | 0 | 0 | 0 | 250 | 0 | 250 |
| 500 | 437 | 0 | 0 | 0 | 500 | 0 | 500 |
| 750 | 625 | 0 | 187 | 0 | 750 | 0 | 750 |
| 1000 | 1120 | 0 | 437 | 0 | 1000 | 937 | 1000 |
| 1250 | 2680 | 0 | 1310 | 0 | 1120 | 1120 | 1120 |
| 1500 | 4310 | 0 | 2500 | 0 | 1250 | 1250 | 1250 |
| 1750 | 7930 | 0 | 6300 | 0 | 1370 | 1370 | 1370 |
| 1870 | 7930 | 0 | 7000 | 0 | 1370 | 1430 | 1370 |
| 2060 | 7930 | 0 | 4800 | 0 | 1370 | 1370 | 1370 |
| 2500 | 7930 | 0 | 5830 | 0 | 1430 | 1250 | 1500 |
| 2500 | 7930 | 0 | 7680 | 0 | 1500 | 1500 | 1500 |
| 2930 | 4750 | 0 | 2000 | 0 | 1060 | 1060 | 1060 |

| Resistor Currents | | | Capacitor Currents | | | |
|---|---|---|---|---|---|---|
| $I_{R1}$ | $I_{R2}$ | $I_{R3}$ | $I_{C1}$ | $I_{C2}$ | $I_{C3}$ | $I_{CFF}$ |
| 187 | 0 | 250 | 250 | 187 | 250 | 187 |
| 437 | 125 | 500 | 500 | 437 | 500 | 437 |
| 625 | 187 | 750 | 750 | 625 | 750 | 625 |
| 1200 | 437 | 1000 | 1000 | 1200 | 1000 | 1200 |
| 2680 | 1250 | 1250 | 1250 | 2680 | 1120 | 2680 |
| 4430 | 2560 | 1560 | 1500 | 4370 | 1250 | 2000 |
| 7930 | 6000 | 2060 | 2180 | 7930 | 1370 | 4680 |
| 7930 | 7060 | 2060 | 2250 | 7930 | 1370 | 7930 |
| 7930 | 4750 | 2120 | 2180 | 7930 | 1370 | 7930 |
| 5120 | 5930 | 2250 | 2370 | 7930 | 1870 | 7930 |
| 7930 | 7930 | 2750 | 2680 | 6180 | 1620 | 7930 |
| 4810 | 2120 | 3000 | 2930 | 4870 | 1060 | 4810 |

| Inductor Current | MOSFET Currents | | | Diode Currents | |
|---|---|---|---|---|---|
| $I_{L1}$ | $I_{Q1S}$ | $I_{Q1G}$ | $I_{Q1D}$ | $I_{D1in}$ | $I_{D1out}$ |
| 250 | 0 | 0 | 250 | 250 | 187 |
| 500 | 0 | 0 | 500 | 500 | 437 |
| 750 | 0 | 0 | 750 | 750 | 625 |
| 1000 | 0 | 1000 | 1250 | 1250 | 1120 |
| 1250 | 0 | 1120 | 2870 | 2750 | 2620 |
| 1560 | 0 | 1250 | 4620 | 4680 | 4250 |
| 2120 | 0 | 1370 | 5750 | 5500 | 5060 |
| 2000 | 0 | 1430 | 7930 | 7930 | 5310 |
| 2180 | 0 | 1430 | 7930 | 7930 | 7930 |
| 2370 | 0 | 2600 | 6810 | 6810 | 7930 |
| 2750 | 0 | 1120 | 1120 | 7930 | 7930 |
| 3000 | 0 | 1060 | 3930 | 4930 | 4810 |

| Resistance | | | |
|---|---|---|---|
| $R_1$ | $R_2$ | $R_3$ | $R_L$ |
| 193770 | 64470 | 10.07 | 988 |
| 193770 | 64470 | 10.07 | 988 |
| 193770 | 64470 | 10.07 | 988 |
| 193770 | 64470 | 10.07 | 988 |
| 193770 | 64470 | 10.07 | 988 |
| 193770 | 64470 | 10.07 | 988 |
| 193770 | 64470 | 10.07 | 988 |
| 193770 | 64470 | 10.07 | 988 |
| 193770 | 64470 | 10.07 | 988 |
| 193770 | 64470 | 10.07 | 988 |
| 193770 | 64470 | 10.07 | 988 |
| 193770 | 64470 | 10.07 | 988 |
| 193770 | 64470 | 10.07 | 988 |

| Capacitances | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $C_{1min}$ | $C_1$ | $C_{1max}$ | $C_{2min}$ | $C_2$ | $C_{2max}$ | $C_3$ | $C_{FFmin}$ | $C_{FF}$ | $C_{FFmax}$ |
| 9.00E-06 | 1.00E-05 | 1.10E-05 | 1.98E-05 | 2.20E-05 | 2.42E-05 | 1.00E-07 | 5.58E-08 | 6.20E-08 | 6.82E-08 |
| 9.00E-06 | 1.00E-05 | 1.10E-05 | 1.98E-05 | 2.20E-05 | 2.42E-05 | 1.00E-07 | 5.58E-08 | 6.20E-08 | 6.82E-08 |
| 9.00E-06 | 1.00E-05 | 1.10E-05 | 1.98E-05 | 2.20E-05 | 2.42E-05 | 1.00E-07 | 5.58E-08 | 6.20E-08 | 6.82E-08 |
| 9.00E-06 | 1.00E-05 | 1.10E-05 | 1.98E-05 | 2.20E-05 | 2.42E-05 | 1.00E-07 | 5.58E-08 | 6.20E-08 | 6.82E-08 |
| 9.00E-06 | 1.00E-05 | 1.10E-05 | 1.98E-05 | 2.20E-05 | 2.42E-05 | 1.00E-07 | 5.58E-08 | 6.20E-08 | 6.82E-08 |
| 9.00E-06 | 1.00E-05 | 1.10E-05 | 1.98E-05 | 2.20E-05 | 2.42E-05 | 1.00E-07 | 5.58E-08 | 6.20E-08 | 6.82E-08 |
| 9.00E-06 | 1.00E-05 | 1.10E-05 | 1.98E-05 | 2.20E-05 | 2.42E-05 | 1.00E-07 | 5.58E-08 | 6.20E-08 | 6.82E-08 |
| 9.00E-06 | 1.00E-05 | 1.10E-05 | 1.98E-05 | 2.20E-05 | 2.42E-05 | 1.00E-07 | 5.58E-08 | 6.20E-08 | 6.82E-08 |
| 9.00E-06 | 1.00E-05 | 1.10E-05 | 1.98E-05 | 2.20E-05 | 2.42E-05 | 1.00E-07 | 5.58E-08 | 6.20E-08 | 6.82E-08 |
| 9.00E-06 | 1.00E-05 | 1.10E-05 | 1.98E-05 | 2.20E-05 | 2.42E-05 | 1.00E-07 | 5.58E-08 | 6.20E-08 | 6.82E-08 |
| 9.00E-06 | 1.00E-05 | 1.10E-05 | 1.98E-05 | 2.20E-05 | 2.42E-05 | 1.00E-07 | 5.58E-08 | 6.20E-08 | 6.82E-08 |
| 9.00E-06 | 1.00E-05 | 1.10E-05 | 1.98E-05 | 2.20E-05 | 2.42E-05 | 1.00E-07 | 5.58E-08 | 6.20E-08 | 6.82E-08 |
| 9.00E-06 | 1.00E-05 | 1.10E-05 | 1.98E-05 | 2.20E-05 | 2.42E-05 | 1.00E-07 | 5.58E-08 | 6.20E-08 | 6.82E-08 |
| 9.00E-06 | 1.00E-05 | 1.10E-05 | 1.98E-05 | 2.20E-05 | 2.42E-05 | 1.00E-07 | 5.58E-08 | 6.20E-08 | 6.82E-08 |

| Inductances | | |
|---|---|---|
| $L_{1min}$ | $L_1$ | $L_{1max}$ |
| 2.64E-07 | 3.30E-07 | 3.96E-07 |
| 2.64E-07 | 3.30E-07 | 3.96E-07 |
| 2.64E-07 | 3.30E-07 | 3.96E-07 |
| 2.64E-07 | 3.30E-07 | 3.96E-07 |
| 2.64E-07 | 3.30E-07 | 3.96E-07 |
| 2.64E-07 | 3.30E-07 | 3.96E-07 |
| 2.64E-07 | 3.30E-07 | 3.96E-07 |
| 2.64E-07 | 3.30E-07 | 3.96E-07 |
| 2.64E-07 | 3.30E-07 | 3.96E-07 |
| 2.64E-07 | 3.30E-07 | 3.96E-07 |
| 2.64E-07 | 3.30E-07 | 3.96E-07 |
| 2.64E-07 | 3.30E-07 | 3.96E-07 |
| 2.64E-07 | 3.30E-07 | 3.96E-07 |
| 2.64E-07 | 3.30E-07 | 3.96E-07 |

| | | MAX1524 Power Losses | | | | | |
|---|---|---|---|---|---|---|---|
| $P_{in}$ | $P_{out}$ | $P_{GND}$ | $P_{FB}$ | $P_{SET}$ | $P_{SHDN}$ | $P_{EXT}$ | $P_{CC}$ |
| 0.078 | 0.034969 | 0 | 0 | 0 | 0.0625 | 0 | 0.0625 |
| 0.25 | 0.190969 | 0 | 0 | 0 | 0.25 | 0 | 0.25 |
| 0.5625 | 0.390625 | 0 | 0 | 0 | 0.5625 | 0 | 0.5625 |
| 1 | 1.2544 | 0 | 0.190969 | 0 | 1 | 0.937 | 1 |
| 1.5625 | 7.0216 | 0 | 1.7947 | 0 | 1.2544 | 1.2544 | 1.2544 |
| 2.25 | 18.8347 | 0 | 6.25 | 0 | 1.5625 | 1.5625 | 1.5625 |
| 3.0625 | 62.8849 | 0 | 22.428 | 0 | 1.7947 | 1.8769 | 1.7947 |
| 3.74 | 62.8849 | 0 | 11.76 | 0 | 1.8769 | 1.9591 | 1.8769 |
| 4.635 | 62.8849 | 0 | 24 | 0 | 1.8769 | 1.8769 | 1.8769 |
| 6.25 | 62.8849 | 0 | 46.2319 | 0 | 2.0449 | 1.7875 | 2.145 |
| 6.875 | 62.8849 | 0 | 53.76 | 0 | 2.145 | 2.145 | 2.145 |
| 8.79 | 23.75 | 0 | 4.36 | 0 | 1.1236 | 1.1236 | 1.1236 |

| Resistor Power Losses | | |
|---|---|---|
| $P_{R1}$ | $P_{R2}$ | $P_{R3}$ |
| 0.034969 | 0 | 0.0625 |
| 0.190969 | 0 | 0.25 |
| 0.390625 | 0.034969 | 0.5625 |
| 1.344 | 0.190969 | 1 |
| 7.0216 | 1.7125 | 1.5625 |
| 18.8275 | 6.4 | 2.34 |
| 52.338 | 24 | 3.8522 |
| 62.8849 | 48.0786 | 4.2848 |
| 62.8849 | 23.75 | 4.4944 |
| 40.6016 | 47.0249 | 5.625 |
| 62.8849 | 56.4616 | 7.205 |
| 16.2097 | 4.4944 | 9 |

| Capacitor Power Losses | | | | | | | |
|---|---|---|---|---|---|---|---|
| $P_{C1min}$ | $P_{C1}$ | $P_{C1max}$ | $P_{C2min}$ | $P_{C2}$ | $P_{C2max}$ | $P_{C3}$ | $P_{CFF}$ |
| 2.81E-07 | 3.13E-07 | 3.44E-07 | 3.46E-07 | 3.85E-07 | 4.23E-07 | 3.13E-09 | 9.76E-10 |
| 1.13E-06 | 1.25E-06 | 1.38E-06 | 1.89E-06 | 2.10E-06 | 2.31E-06 | 1.25E-08 | 5.33E-09 |
| 2.53E-06 | 2.81E-06 | 3.09E-06 | 3.87E-06 | 4.30E-06 | 4.73E-06 | 2.82E-08 | 1.09E-08 |
| 4.50E-06 | 5.00E-06 | 5.50E-06 | 1.24E-05 | 1.38E-05 | 1.52E-05 | 5.01E-08 | 3.50E-08 |
| 7.03E-06 | 7.81E-06 | 8.59E-06 | 6.80E-05 | 7.55E-05 | 8.31E-05 | 5.01E-08 | 2.14E-08 |
| 1.01E-05 | 1.13E-05 | 1.24E-05 | 1.84E-04 | 2.04E-04 | 2.25E-04 | 6.97E-08 | 1.04E-07 |
| 1.38E-05 | 1.53E-05 | 1.68E-05 | 6.23E-04 | 6.92E-04 | 7.61E-04 | 8.59E-08 | 2.95E-07 |
| 1.80E-05 | 2.00E-05 | 2.20E-05 | 6.23E-04 | 6.92E-04 | 7.61E-04 | 9.39E-08 | 6.29E-07 |
| 2.14E-05 | 2.38E-05 | 2.61E-05 | 6.23E-04 | 6.92E-04 | 7.61E-04 | 9.39E-08 | 1.75E-06 |
| 2.66E-05 | 2.95E-05 | 3.25E-05 | 6.23E-04 | 6.92E-04 | 7.61E-04 | 1.02E-07 | 1.75E-06 |
| 2.81E-05 | 3.13E-05 | 3.44E-05 | 6.23E-04 | 6.92E-04 | 7.61E-04 | 1.13E-07 | 1.75E-06 |
| 4.05E-05 | 4.50E-05 | 4.95E-05 | 2.41E-04 | 2.67E-04 | 2.94E-04 | 5.62E-08 | 6.78E-07 |

| MOSFET Power Losses | | | Diode Power Losses | | Inductor Power Losses | |
|---|---|---|---|---|---|---|
| $P_{Q1S}$ | $P_{Q1G}$ | $P_{Q1D}$ | $P_{D1in}$ | $P_{D1out}$ | $P_{LR}$ | $P_{Wh}$ |
| 0 | 0 | 0.0625 | 0.0625 | 0.034969 | 0.0625 | 1.03E-02 |
| 0 | 0 | 0.25 | 0.25 | 0.190969 | 0.25 | 4.13E-02 |
| 0 | 0 | 0.5625 | 0.5625 | 0.390625 | 0.5625 | 9.28E-02 |
| 0 | 1 | 1.5625 | 1.5625 | 1.2544 | 1 | 1.65E-01 |
| 0 | 1.2544 | 8.2369 | 7.37 | 6.8644 | 1.5625 | 2.58E-01 |
| 0 | 1.5625 | 20.1894 | 21.06 | 18.5725 | 2.4336 | 4.02E-01 |
| 0 | 1.8769 | 31.2225 | 29.205 | 24.3386 | 3.71 | 7.42E-01 |
| 0 | 1.9591 | 47.0249 | 46.5491 | 27.5058 | 4 | 6.60E-01 |
| 0 | 1.9591 | 62.8849 | 62.8849 | 57.4925 | 4.36 | 7.84E-01 |
| 0 | 3.718 | 54.0033 | 54.0033 | 62.8849 | 5.7591 | 9.27E-01 |
| 0 | 1.6016 | 7.84 | 62.8849 | 62.8849 | 7.04 | 1.25E+00 |
| 0 | 1.1236 | 14.9733 | 24.9458 | 22.8475 | 8.61 | 1.49E+00 |

| **Efficiency** |
| --- |
| η |
| 0.001838884 |
| 0.004338625 |
| 0.003644317 |
| 0.009243811 |
| 0.041637706 |
| 0.080139539 |
| 0.200493194 |
| 0.164610127 |
| 0.132288869 |
| 0.09764896 |
| 0.087837891 |
| 0.023640144 |